Licensed Material — Property of IBM

LY33-8042-6 File No. S370-33 (OS/VS)

Program Product

OS/VS Sort/Merge Logic

Program Number 5740-SM1

Release 5



Seventh Edition (September, 1979)

This edition, as amended by technical newsletters LN20-9345 and LN20-9390, applies to Release 5, Modification 0 of OS/VS Sort/Merge, Program Product 5740-SM1, and to any subsequent releases until otherwise indicated in new editions or technical newsletters. Technical newsletter LN20-9329 is obsolete.

The changes for this edition are summarized under "Summary of Amendments." Specific changes are indicated by a vertical bar to the left of the change. These bars will be deleted at any subsequent republication of the page affected. Editorial changes that have no technical significance are not noted.

Changes are periodically made to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest IBM System/370 and 4300 Processors Bibliography, GC20-0001, for the editions that are applicable and current.

It is possible that this material may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country.

Publications are not stocked at the address given below; requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for reader's comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, P.O. Box 50020, Programming Publishing, San Jose, California, U.S.A. 95150. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Licensed Material--Property of IBM Page of LY33-8042-6, As Updated 31 March 1981, By TNL LN20-9345

SUMMARY OF AMENDMENTS FOR LY33-8042-6 5740-SM1

RELEASE 5, MODIFICATION 0

- Improved performance when sorting variable-length records with new disk sorting technique (VLR-Blockset).
- Ability to add to or change installed or passed user options, using the new OPTION control statement.
- Support of the IBM 3375 direct access storage device for initial input, final output, and intermediate work data sets.
- Ability to produce statistical data about sort applications executed via the System Management Facility (SMF).
- Ability to specify that format CH be translated the same as format AQ.
- Ability to specify that record counters should or should not be checked at the end of execution of sorting applications that use the E35 exit routine without a SORTOUT data set.

SUMMARY OF AMENDMENTS FOR LY33-8042-6 5740-SM1

RELEASE 4

- A further standard disk sorting technique (Blockset) added to improve performance
- The design point is changed from 32K bytes to 48K bytes
- SORTIN/SORTOUT I/O handling is enhanced to improve performance.
- The default printing of the Sort specially formatted dump is removed.
- Support of the IBM 3380 direct access storage device for initial input, final output, amd intermediate work data sets.

SUMMARY OF AMENDMENTS FOR LY33-8042-5 5740-SM1

RELEASE 3

NEW DISK SORTING TECHNIQUE

A new technique, Vale, has been introduced. Its primary function is to extend the benefits of the Peerage technique to variable-length records.

WORK DATA SETS

- The sort program's work data sets can be on a mixture of any of the supported disk types.
- If necessary, a secondary storage allocation is

- automatically made; this need not be specified in JCL.
- Unused space is automatically released; this need not be specified in JCL.
- Work I/O handling has been changed to give improved performance.

LONGER INPUT RECORDS FOR DISK SORT

Input records can be up to 32,760 bytes long, regardless of whether work data sets are on tape or disk.

Preface

This publication describes the functions and internal logic of the OS/VS Sort/Merge Program Product, Program Number 5740-SM1. Its purpose is to aid IBM Program Service Representatives to understand the program and to locate and fix program faults. The reader should be familiar with the information contained in related publications listed below.

This publication has six major parts:

<u>Section 1</u>. Introduction describes the program's structure and operational and physical characteristics and relates it to the operating system.

<u>Section 2</u>. Method of Operation contains method-of-operation diagrams and extended descriptions which detail the program's functions and relate them to specific code modules.

<u>Section 3</u>. Program Organization gives the physical organization of the program in main storage, as well as describing the interface between modules, and record movement.

<u>Section 4</u>. Directories serves as a cross-reference between parts of the program and their description in the manual.

<u>Section 5</u>. Data Areas gives the layout of important data areas used by the program.

<u>Section 6</u>. Debugging Aids gives diagnostic and debugging hints under twelve headings, including how to decide whether a problem is due to a program error, how to add a temporary change to a program module, the origin of error messages, and how to effect simple bypasses.

This publication also contains four appendixes:

Appendix A. Program Exits

Appendix B. Format Codes (Conventional Techniques)

Appendix C. Checkpoint/Restart Facility

Appendix D. Program Listing Standards and Conventions

RELATED PUBLICATIONS

OS/VS Sort/Merge Programmer's Guide, SC33-4035

OS/VS Sort/Merge Installation Guide, SC33-4034

Licensed Material -- Property of IBM Page of LY33-8042-6, As Updated 31 March 1981, By TNL LN20-9345

Contents

	SECTION 1. INTRODUCTION		•	•	. 1
	Relationship to the Operating System				. 1
	Program Structure				. 2
	Tape Sort Techniques	-		•	. 2
	Disk Sort Techniques				. 2
-1	Conditions for Use of Blockset Sorting Techniques				2.1
٠	Operational Considerations				
	Physical Characteristics	•		•	7
	Main Storage	•	•	•	• ;
	Module Residence	•	•	•	• ;
	Modute restrence	•	•	•	• '
	SECTION 2. METHOD OF OPERATION				10
	Section 2. Method of Operation	•	•	•	- 10
	How the Section is Organized	•	•	•	- 10
	How to Use the MO Diagrams	•	•	•	- 10
	References to Data Areas				
	Symbols Used	•	•	•	.11
	How to Use the Module Tables	•	•	•	. 12
	Abbreviations Used in Module Tables	•	•	•	. 12
	SECTION 3. PROGRAM ORGANIZATION	•	•	•	. 85
	Program Exit Handling	•	•	•	. 85
	Module Interface	•	-	•	.85
	Phase 0	•	•	•	.89
	Phase 1				
	Phase 2	•	•		.89
	Blockset Technique				.89
	Other Techniques				.90
	Phase 3				.91
	Phase Structures				.91
	Storage Layouts				104
	Blockset, Peerage, and Vale Techniques	_			104
	Other Techniques	-	•	-	104
		•	•	•	
	SECTION 4. DIRECTORIES			_	109
	Blockset Directory				
	Peerage and Vale Directory	•	•	•	111
	Directory for Conventional Techniques	•	•	•	112
	Directory for Conventional Techniques	•	•	•	112
	Explanacion of Column Headings	•	•	•	112
	SECTION 5. DATA AREAS				120
	Blockset Area (COMMON)	•	•	•	120
	COMMON Princest	•	•	•	121
	COMMON Printout	•	•	•	121
	Peerage and Vale Communication Area (COMMA)	•	•	•	129
	Control Phase Information Area (CPI), Conventional Techniques. Phase-to-Phase Information Area (PPI), Conventional Techniques	•	•	•	139
	Phase-to-Phase Information Area (PPI), Conventional Techniques	•	•	•	141
	Index to PPI	•	•	•	152
	Module ICEAM1: Generated Defaults	•	•	•	157
					. . .
	SECTION 6. DEBUGGING AIDS	•	•	•	159
	Defining Problem Cause	•	•	•	159
	Is This a Program Error?	•	•	•	160
	Potential Problems with Routines at Program Exits				
	Potential Problems With Invoking Programs	•	•	•	161
	Considerations When an Error Has Been Located				
	When Not to Waste Time on Repair	•	•		162
	Bypassing a Problem	•	•		162
	Reporting a Problem		•		163
	Microfiche Organization		_		164
	Adding a Temporary Change to the Maintenance Area	•			165
	How Sort Uses Registers	•			165
	Finding DCBs and IOBs for SORTIN, SORTOUT, and SORTWK	_	_	_	166

Licensed Material--Property of IBM Page of LY33-8042-6, As Updated 31 March 1981, By TNL LN20-9345

Blockset Technique							166
Peerage and Vale Techniques							168
The DEBUG Control Statement							172
Syntax			•				173
Messages Produced by Using the DEBUG Control Statement							175
Messages Produced by Using the DIAG Option							177
Dumps							178
Normal ABEND Dumps		_				_	178
The TRACE Table		-		_	_	_	178
Forcing a Specially Formatted Dump	-	_	-	_			187
Finding an Object Module in a Storage Dump							
Origin of Program Messages	•	•	•	•	•	•	191
Blockset Technique	•	•	•	•	•	•	191
All Other Techniques	•	•	•	•	•	•	192
Cross-Reference Tables	•	•	•	•	•	•	196
CPI-PPI Cross-Reference Table	•	•	•	•	•	•	196
Common Cross-Reference Table (BLOCKSET)	•	•	•	•	•	•	301
COMMA Cross-Reference Table (Peerage/Vale)	•	•	• ,	•	•	•	204
comme cross-vererence rable (reetage/vare)	•	•	•	•	•	•	203
APPENDIX A. PROGRAM EXITS							210
Calling Modules							
Register Usage	•	•	•	•	•	•	210
Register usage	•	•	•	•	•	•	219
APPENDIX B. FORMAT CODES (CONVENTIONAL TECHNIQUES)							220
Fixed-length Records	•	•	•	•	•	•	220
Variable-length Records	•	•	•	•	•	•	221
Variable-length Records	•	•	•	•	•	•	221
Format Condition Codes	•	•	•	•	•	•	222
APPENDIX C. CHECKPOINT/RESTART FACILITY							224
APPENDIA C. CHECKPOINT/RESTART FACILITY	•	•	•	•	•	•	224
Sorting Application (Peerage and Vale)	•	•	•	•	•	•	224
Sorting Application (Conventional Techniques)	•	•	•	•	•	•	225
Merge-Only Application	•	•	•	•	•	•	225
APPENDIX D. PROGRAM LISTING STANDARDS AND CONVENTIONS							226
Blockset							
Module Names							
Instruction Names	•	•	•	•	•	•	226
Constant Names	•	•	•	•	•	•	226
Work Area Names							
Peerage and Vale	•	٠	•	•	•	•	226
Module Names	•	-	•	•	•	•	226
Instruction Names							
Constant Names							
Work Area Names							
Conventional Techniques	•	•	•	•	•	•	227
Module Names							
Instruction Names	•	•	•	•	•	•	227
Constant Names	•	•		•	•		228
Work Area Names							
Table Names							
PPI Area Names	•		•	•	•		228
Use of Routines in More Than One Module				•			228
·							
APPENDIX E. SMF RECORD DSECT (ICESMF)					•		228.1
TNDEV							220

Figures

Figure	1.	Phases of the Program	4
Figure		Initiation of the Program	5
Figure		Initiation Via the ATTACH Macro	6
Figure	4.	System Libraries	7
Figure	5.	Data Sets Used Only During Execution	9
Figure	6.	Table of Contents for Method of Operation Diagrams .	13
Figure		Load Module Interface	86
Figure	8.	Peerage Modules	92
		Vale Modules	93
		FLR-Blockset Modules	94
		.VLR-Blockset Modules	
		Conventional Technique Modules	
		Object Modules in the Two Overlay Load Modules	97
		Load Modules with Conventional Techniques	
		Storage Layouts for Conventional Techniques	10
		Map of PPI	142
		A Sample Set of Messages	160
		Microfiches for Load Module ICEMESI	164
		Use of Registers by Standard Disk Techniques	16
		Locating Control Blocks in a Dump with Blockset	16
Figure	20.	Locating SORTIN and SORTOUT Control	
_		Blocks with EXCP	169
		Locating Control Blocks in a Dump with Peerage/Vale.	17
		Contents of a Specially Formatted Dump	18
		Interpreting a Formatted Dump	189
Figure	24.	The Start of a Module in a Dump	190

Section 1. Introduction

The OS/VS Sort/Merge Program Product has three standard components:

- Disk sort, which is reenterable
- Tape sort
- Merge only

In addition it contains two conventional disk sorting techniques, normally not used.

The total program is a generalized sort/merge program working as a processing program under the operating system. It can sort one VSAM or QSAM data set, and can merge up to 16 VSAM or QSAM data sets. Each of the QSAM data sets may consist of a number of concatenated data sets, up to the system maximum.

Input and output may be fixed or variable length, blocked or unblocked, and may be on any device supported by QSAM or VSAM. Output may be in VSAM form even if input is QSAM, as long as the output data set has previously been defined using the AMS utility.

Most sorting applications need intermediate (work) storage, which may be on:

- 2400 or 3400 series tapes
- 2314 or 2319 direct access devices
- 3330 series direct access devices (3330/3333, Model 1 and/or Model 11)
- 3340 direct access devices
- 3350 direct access devices
- 3375 direct access devices
- 3380 direct access devices
- 3380 direct access devices and 3880 Models 2 or 3 with Speed Matching Buffer
- 3380 direct access devices and 3880 Model 13 with Cache
- 3850 MSS volumes

Direct access device types can be mixed.

Relationship to the Operating System

The program communicates through macro instructions and interruptions with the operating system control program.

For example, for input and output the data management EXCP, QSAM, BSAM, or VSAM routines can be used; the EXCP (or EXCPVR, SVS, and VS1 only) macro instruction is used for intermediate storage I/O operations, and sometimes for input and output (EXCPVR is never used for input or output); and the linkage editor may be called to link-edit user-written routines to the program.

The program is designed to make use of the following facilities if they are included at system generation: Input/Output Recovery Management Support; Multiple Console Support; and (at the user's request) Checkpoint/Restart.

Program Structure

The program operates in four major phases, as indicated in Figure 1.

The first part of Phase 0 will be covered here. This part reads and interprets program control statements, and selects the technique to be used.

For a merge, there is only one technique.

TAPE SORT TECHNIQUES

If tape is specified for work areas, one of the conventional tape sorts (BALN, POLY, or OSCL) is used. The choice of technique depends solely on the numbers of work and input tapes specified.

DISK SORT TECHNIQUES

There are four standard disk sorting techniques available to the sort/merge program:

- FLR-Blockset--fixed-length records
- VLR-Blockset--variable-length records
- Peerage--fixed-length records
- Vale--both fixed- and variable-length records

Sort/merge will select one of the Blockset techniques if all the conditions for its use are met (see "Conditions for Use of Blockset Sorting Techniques").

Disk Sorting Techniques For Fixed-Length Records

The sort/merge program's most efficient fixed-length record technique, FLR-Blockset, will be used for most sorting applications, providing the conditions listed in "Conditions for Use of Blockset Techniques" are met. If one or more of the conditions for the FLR-Blockset technique are not met, the Peerage or Vale technique will be used, where possible.

Disk Sorting Techniques For Variable-Length Records

The high-performance VLR-Blockset technique will be used for sorting variable-length records if all the requirements listed in the following section are fulfilled. If not, the current variable-length disk sorting technique, Vale, will generally be used.

To enable sort/merge to attempt to select the best technique, whether VLR-Blockset or Vale, the following guidelines may be useful. If the average length of variable records is more than 350 bytes, you should specify the L5 operand on the RECORD control statement. If you specify an L5 operand that is between 350 and 1,000 bytes, Sort uses the Vale technique when the ratio of region size to number of records is large. When L5 is greater than 1,000 bytes, Vale is generally used. If the

working storage is less than 100K bytes, sort/merge will attempt to select VLR-Blockset regardless of average record length. If you don't specify L5, sort/merge will try to use VLR-Blockset.

When used, the new VLR-Blockset technique will generally show processing time improvement over Vale.

CONDITIONS FOR USE OF BLOCKSET SORTING TECHNIQUES

The sort/merge program has two high-performance disk sorting techniques, FLR-Blockset and VLR-Blockset, for fixed- and variable-length records, respectively. The program will first attempt to use one of these techniques, providing the following conditions are fulfilled. If they are not, then one of the other standard disk sorting techniques, Peerage or Vale, may be used where possible (Peerage or Vale for fixed-length records; Vale for variable-length records).

The first list below includes conditions common to both techniques. The second list includes conditions relevant to FLR-Blockset only, and the third, to VLR-Blockset only.

Conditions Common To Both Blockset Techniques

- More than about 64K bytes of main storage plus additional storage for buffers are available for sort and other possible modules in the region/partition. The larger the input/output block sizes are, the larger main storage must be.
- No program exits other than E15 and/or E35 (without overlay structures) provided they are pre-linkedited.
- If a SORTCNTL DD statement is used, no control statements other than OPTION should be included.
- Tape work data set is not specified.
- Under MVS, up to 26 dynamically allocated sort work data sets may be used, depending on the complexity of the control field and use of SMF.
- Input or output is not a VSAM or an ASCII data set, or track overflow record format (RECFM=FT).
- Input is not a direct-access data set with key sequenced organization (BDAM).
- Input or output must not be a spool or a dummy data set.
- Output cannot be padded or truncated records, or an old data set residing on tape.
- Multivolume disk data output is not requested.
- Checkpoint is not specified.
- Control fields that do not exceed 248 bytes.
- Control fields that do not cause the intermediate record to expand by more than 30% of the total record length. Factors that might expand the record are overlapping fields, decimal fields, fields that require translation, or specification of EQUALS.
- All supported control field formats except those with leading, trailing, overpunched, or separate signs, or ASCII format control fields.
- Skipping of input records is not requested.

FLR-Blockset Conditions

- SORTIN record length plus 13 bytes and any additional bytes caused by control field expansion must not exceed the smallest SORTWK track capacity or 32K bytes, whichever is smaller.
- Record length is not to be changed by program exit E15 and/or E35.
- SORTWK data sets must be allocated in cylinders (MVS only).

VLR-Blockset Conditions

VLR-Blockset minimum storage requirements are defined by the following computations (whichever results in the larger value should be used, but in no case should less than 69K bytes be used):

- In computing the amount of storage necessary to execute VLR-Blockset, use whichever of the following computations resulting in the largest value:
 - 48K bytes of main storage plus the largest of three times:
 - a. the maximum input block size, or
 - b. the maximum output block size, or
 - c. 2000 bytes.
 - 2. 48K of main storage plus four times the size of the maximum record length, plus the largest of the following:
 - a. the maximum input block size, or
 - b. the maximum output block size, or
 - c. 2000 bytes.
- Maximum record length does not exceed the track length for the SORTIN or SORTOUT disk data set, or 32,000 bytes, whichever is smaller.
- Input or output is not spanned, variable-length records.
- Input or output is not Format D records (variable-length ASCII tape records).
- Work data sets are specified (a sort in main storage is not supported).
- The sort/merge program is not dynamically invoked by IMS/VS for variable-length record sorting applications.
- The control field does not overlap the record descriptor word (RDW).
- If the ratio of region size to the number of input records is large, and if the L5 operand specified on the RECORD control statement is greater than 350 bytes, sort/merge may, in some cases, choose to use the Vale technique. If L5 is not specified, sort/merge will execute VLR-Blockset if all other conditions are met.

Program Structure

As shown in Figure 1, the program operates in four major phases. The first part of Phase 0 reads and interprets program control statements, and selects the technique to be used.

| For a merge there is only one technique.

| TAPE SORT TECHNIQUES

| If tape is specified for work areas one of the conventional tape sorts | (BALN, POLY, or OSCL) is used. The choice of technique depends solely on the numbers of work and input tapes specified.

| DISK SORT TECHNIQUES

| For a disk sort the program selects one of three standard techniques: | BLOCKSET, PEERAGE, or VALE.

| The program's most efficient technique, BLOCKSET will be used for most | sorting applications fulfilling the following conditions:

- Fixed-length records up to the length of the smallest SORTWK disk
 track size
- More than about 60K bytes of main storage available for sorting and possible modules in the region/partition
- No program exits other than E15 and/or E35, provided they are prelink-edited
- Input or output is <u>not</u>:
- VSAM data set
- ASCII data set
- Data set with track overflow record format (RECFM=FT)
- Direct access keyed sequenced data set
- Residing on a device type not supported as a sort work area
- Spool data set or dummy data set

If these conditions are not met, the program will attempt to use Peerage or Vale. Peerage is normally used if the following criteria are met:

- Fixed-length records
- Record length no greater than track length
- No exits to be activated other than E15, E18, E35, E39, or E61
- Control word not too long (see note)

Note: No figure can be given for how long the control word can be if the Peerage technique is to be used; it depends on many variables, such as device type for work storage and amount of main storage available for buffers. However, the length limit is unlikely to be reached before 256 bytes, and will usually be considerably higher.

If any one of the conditions mentioned above is not satisfied, the program will attempt to use Vale.

The user should not normally need to be aware that these different techniques exist. In the OS/VS Sort/Merge Programmer's Guide, they are collectively referred to as the "standard disk technique." However, it is possible to specify, when the program is installed or in the OPTION or DEBUG statement on execution, that a Blockset technique should not be used.

An informational message (ICE092I or ICE093I) states which of the standard disk techniques has been used.

The conventional disk sorts supplied with the program (BALN and CRCX) can be forced by a parameter of the DEBUG statement. Care should then be taken that the SORTWK requirements for the forced technique have been met. For information on these requirements, refer to the OS/VS Sort/Merge Programmer's Guide.

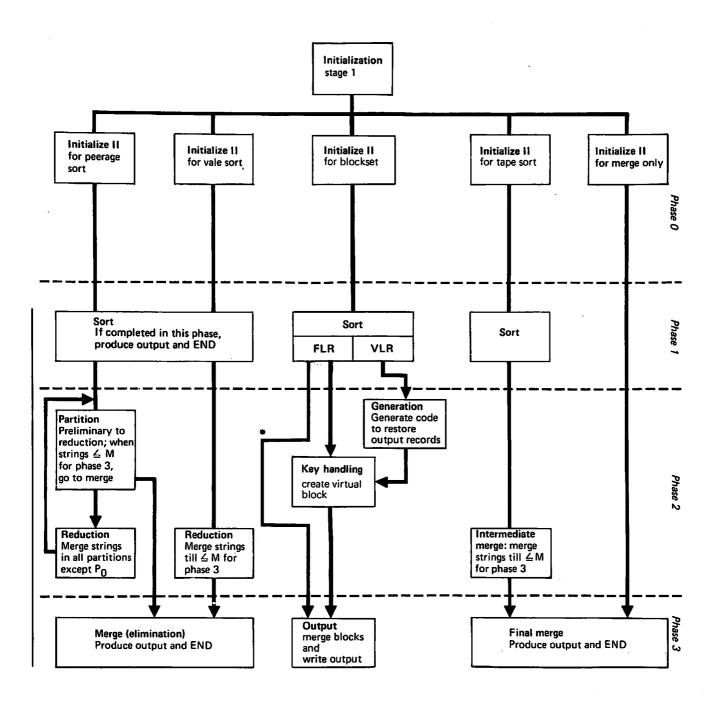


Figure 1. Phases of the Program

^{*}If all records fit in main storage or just 1 string of data written on SORTWKs, the key handling is bypassed.

Operational Considerations

The program may be initiated by an EXEC statement or invoked from a processing program, as shown in Figure 2.

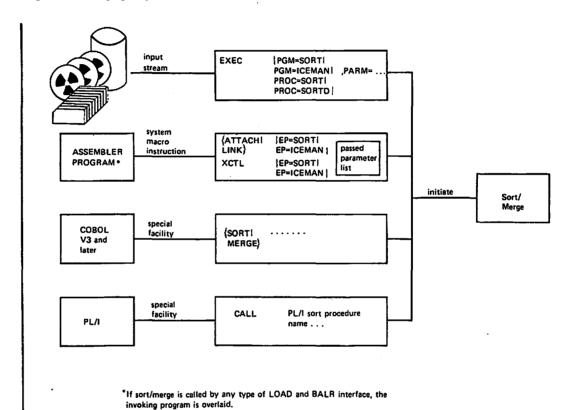


Figure 2. Initiation of the Program

For an EXEC sort or merge, control information is supplied via the following control statements in SYSIN: SORT, MERGE, OPTION, RECORD, MODS, ALTSEQ, DEBUG, and END.

For a sort or merge invoked from an assembler-language program, control information is supplied in a parameter list and SORTCNTL. For COBOL and PL/I, additional information may be passed via the SORTCNTL data set.

Figure 3 gives an example of the use of the ATTACH macro instruction.

User routines for use at program exits may be located in partitioned data sets defined by the user in the MODS statement, or may be in SYSIN, in which case they are copied into the SORTMODS data set. The user defines link-editing requirements: none, separate, or together with other routines for the same phase.

	LA ATTA	1,PARLST CH EP=SORT	LOAD ADDR OF PARAM POINTER IN R1 INVOKE SORT
PARLST	DC .	X°80°,AL3 (ADLST)	POINTER FLAG/ADDRESS OF PARAM LIST
ADLST LISTBEG	CNOP DC DC DC	2,4 AL2 (LISTEND-LISTBEG) A (SORTA)	ALIGN TO CORRECT BOUNDARY PARAM LIST LENGTH BEGINNING ADDRESS OF SORT STMT
	DC DC DC DC DC	A (RECA) A (RECZ) A (MOD1) A (MOD2) C'ABC#'	ALIGN TO CORRECT BOUNDARY PARAM LIST LENGTH BEGINNING ADDRESS OF SORT STMT END ADDRESS OF SORT STMT BEGINNING ADDR OF RECORD STMT END ADDR OF RECORD STMT ADDR OF E15 RTN ADDR OF E35 RTN DDNAME CHARACTERS OPTIONAL MAIN STORAGE FLAG BYTE OPTIONAL MAIN STORAGE IN HEX MESSAGE OPTION CLARACTERS OPTIONAL MAIN STORAGE IN HEX MESSAGE OPTION CLARACTERS OPTIONAL MAIN STORAGE IN HEX
LISTEND	DC DC DC	X'00' X'01A000' X'FF' C'(U)'	OPTIONAL MAIN STORAGE FLAG BYTE OPTIONAL MAIN STORAGE IN HEX MESSAGE OPTION FLAG BYTE MESSAGE OPTION
SORTA	DC	C' SORT FIELDS= (10,15, C'FILSZ=4780'	CH,A), SORT CONTROL STMT (CONTINUED)
SORTZ		-	DELIMITER
RECA	DC	C' RECORD LENGTH=100,1	YPE=F' RECORD CONTROL STMT
RECZ MOD1	DS	C' ' OH G *,15	DELIMITER (routine for exit E15)
MOD2	DS USING	ОН G *,15	(routine for exit E35)

Figure 3. Initiation Via the ATTACH Macro

Physical Characteristics

When a standard disk technique is used, the program is reenterable.

MAIN STORAGE

The program needs a minimum of 48K bytes of main storage. If performance is a consideration you should allocate at least 72K bytes, and preferably between 128K bytes and 512K bytes.

The program will make use of all available main storage in the partition or region if SIZE (MAX) is in effect; this can be specified either at installation or at execution. The MAXLIM parameter should have been specified at installation time: this sets an upper limit to the amount of space which the program can use. If it has not been specified, and SIZE (MAX) is used in a very large virtual partition or region, performance will usually deteriorate and deactivation may result.

LIBRARIES

There are two link libraries designated for your sort/merge program-one containing the reenterable routines, and the other containing non-reenterable initiating routines. In addition, a sort library containing all other sort/merge routines as well as space in the system SVC library is designated when you are using any sort application with tape work areas or any merge application, or if any of the nonstandard disk techniques are forced. If sort/merge cataloged procedures are used, space in a procedure library is also designated.

System Libraries

The system libraries usually used are shown in Figure 4. In addition, Figure 5 shows how the data sets are used during execution of sort/ merge.

System Libraries	VS1 (also MVT, MFT)	SVS, MVS
Link library (reenterable modules)	SYS1.LINKLIB	SYS1.LPALIB
Link library (non- reenterable modules)	SYS1.LINKLIB	SYS1.LINKLIB
Sort library	SYS1.SORTLIB	SYS1.SORTLIB
SVC library	SYS1.SVCLIB	SYS1.LPALIB
Procedure library	SYS1.PROCLIB	SYS1.PROCLIB

Figure 4. System Libraries

If SYS1.PROCLIB is used for procedures, any earlier SORT or SORTD procedures (SM-023 or 5734-SM1) should be scratched.

Private Libraries

If you have more than one operating system and use private libraries, only one set of modules is needed (except for the SVC routine, which is dependent on the system under which you are running; it must always be in a system library).

DATA SETS

Temporary data sets are used at execution time. These are illustrated in Figure 5.

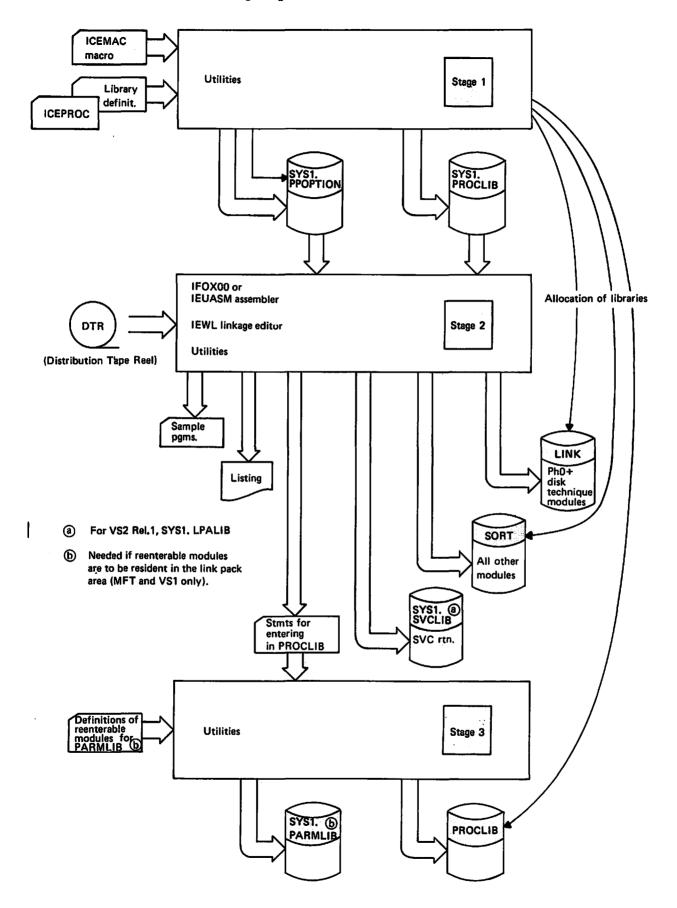


Figure 4. The Installation Procedure

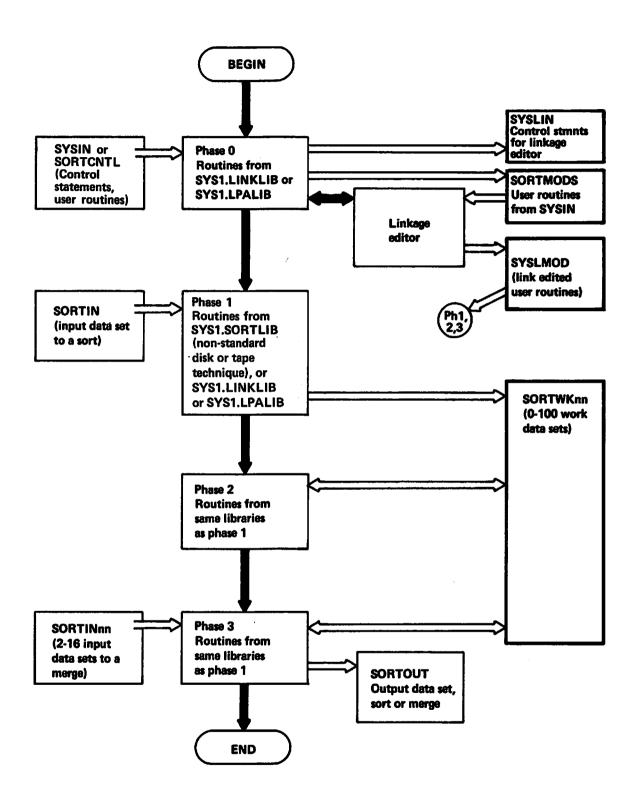


Figure 5. Data Sets Used Only During Execution

Section 2: Method of Operation

This section gives a functional description of the sort/merge program, with cross references to other parts of the manual. It also provides an entry into the microfiche by showing which modules perform which function. It does not, however, describe in detail the various techniques used by the program.

How the Section Is Organized

The section contains information in four forms:

- A hierarchical overview of the functions of the program, Figure 6, which also serves as a table of contents to the diagrams.
- 2. The Method of Operation (MO) diagrams, each describing a function of the program in the form input-process-output.
- 3. Extended descriptions accompanying each lowest-level diagram.
- 4. Module tables accompanying each lowest-level diagram for conventional techniques, showing which modules perform which function. For peerage and vale modules this information is included in the extended descriptions.

How to Use the MO Diagrams

To find the diagram you need, refer to the overview (Figure 6).

The diagrams are arranged with the processes (functions) listed and numbered in the central block. Start reading in that block, and refer to the left-hand column to find input to each function, and the right-hand column for output. The extended descriptions give more detailed information.

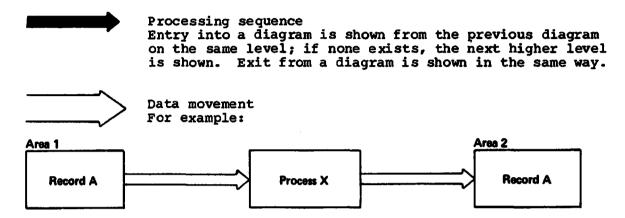
REFERENCES TO DATA AREAS

| The program makes extensive use of a large data area. For Blockset, the | area is called COMMON. See Section 5 for the area layout. For Peerage | and Vale appplications the area is called COMMA, and its layout can be | found in Section 5. For other applications the information in COMMA is moved in Phase 0 to a 'control phase information area', the CPI, and thence to a 'phase-to-phase information area', the PPI, the DSECT for which resides in module ICERCA.

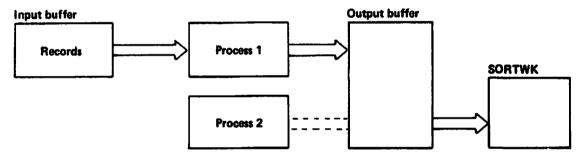
Virtually all parts of the program use COMMON, COMMA, or the PPI. This is not therefore normally shown in the MO diagrams; exceptions are made when for example updates to COMMON, COMMA, or the PPI, form the major output from a function.

The data areas themselves are described in detail in Section 5 of this publication. To find which specific fields are referenced by a particular CSECT, see the cross-reference list in the relevant | microfiche. To find which modules reference a specific field, see the cross-reference tables in Section 6.

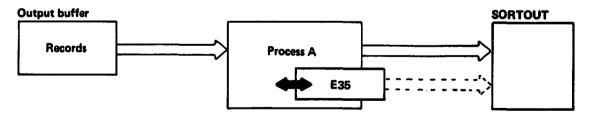
SYMBOLS USED



Process X moves record A from area 1 to area 2.



Process 1 moves records from an input buffer to an output buffer. Process 2 moves them from the output buffer to a SORTWK data set.



Process A moves records from an output buffer to the SORTOUT data set. It also passes control to any user routine present at exit E35, which may also move records to SORTOUT. ----

Pointer - the field from which the arrow originates contains the address of the field or area to which the arrow points. This symbol is also used in the extended descriptions to the diagrams, as follows: '...Exit Name Table (*PPIAPGC means that a pointer to the Exit Name Table can be found in PPIAPGC.



Data Reference - the process block from which the arrow originates accesses the information in the area to which the arrow points, without modifying it.



Diagram Reference - the functions in the process block are described in greater detail in diagram N.

How to Use the Module Tables

Function

Relates to a function in the process column of the relevant diagram. To find which module carries out the function, follow across to the next column.

Conditions

Work from left to right choosing the conditions which apply to your application.

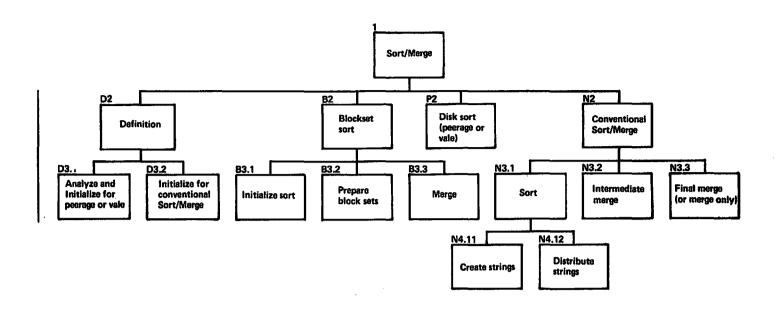
Mod

Gives the last three or four letters of the name of the module which carries out the function. A directory of modules is given in Section 4.

When a running module carries out a function it alone is referenced, though the corresponding assignment module is of course also necessary to the function. When an assignment module actually carries out the function, it alone is referenced. See Section 4 for more information on assignment and running modules.

ABBREVIATIONS USED IN MODULE TABLES

Fix	fixed-length records
Var	variable-length records
Exit	one or more program exits activated
No E	no program exits activated
256+	records of 256 or more bytes
<256	records of less than 256 bytes
Mult	a multiple compare routine used
Extract	an extract compare routine used
Spanned	output consists of spanned records
Forward	(forwd) read tape forward
Backward	(back) read tape backward



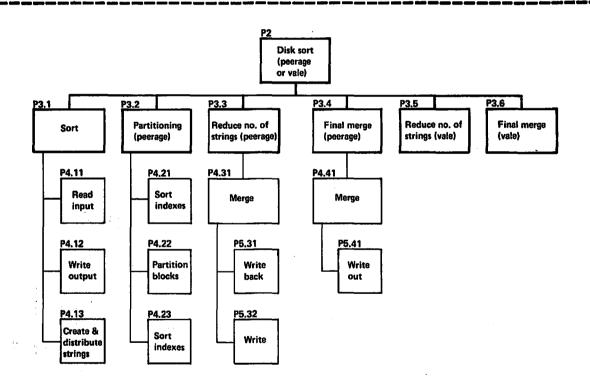
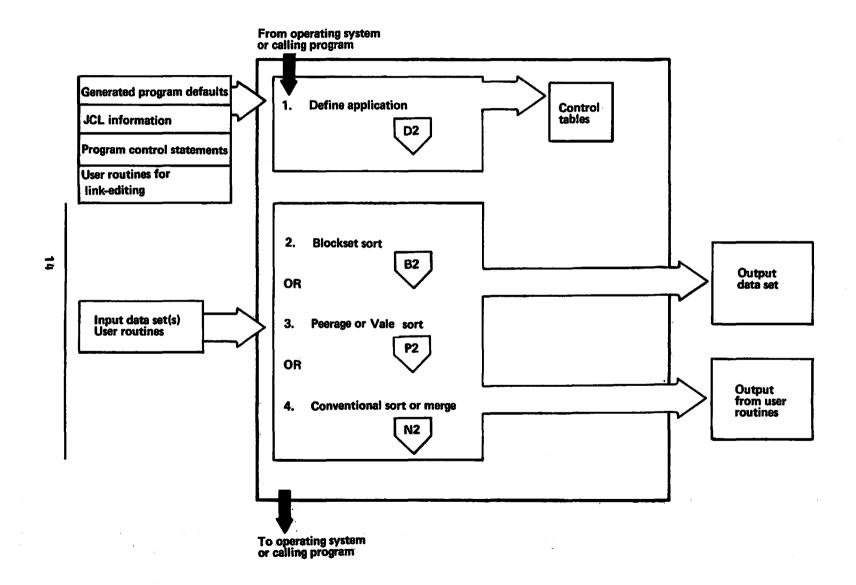


Figure 6. Table of Contents for Method of Operation Diagrams Hierarchical Overview of Program Functions



Licensed Material--Property of IBM

Extended Description for Diagram 1

1. Analyze control information

Determine whether the application is a Blockset, Peerage, or Vale technique sort, or a conventional technique Sort or Merge.

Initialize for the appropriate application.

| 2. Blockset sort

Sort input into ordered strings, attempting to limit the key range of cylinders.

Create optimal virtual block sets from physical blocks on work file cylinders.

Merge virtual blocks to produce final output.

3. Peerage or Vale sort

Sort input into ordered strings, using the replacement selection technique.

For Peerage, partition the work files. The partitioning technique is shown in MO diagram P3.2.

Reduce the number of strings.

Merge the strings to produce final outrut.

4. Conventional Sort

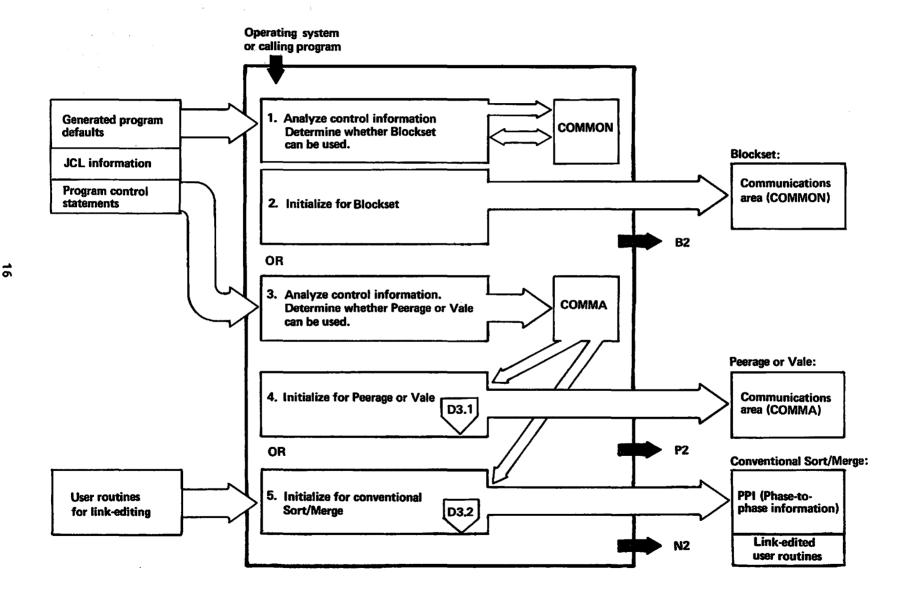
Sort input into ordered strings, using the replacement selection technique.

Reduce the number of strings.

Merge the strings to produce final output.

Merge

Merge the input files to produce final output.



Extended Description for Diagram D2

 Process program options (default and specified), JCL information, and program control statements.

Optimize control field specifications.

Check Blockset technique criteria (listed in Section 1 under 'Program Organization').

2. Initialize for Blockset:

Generate code to handle control fields.

Make the program reenterable.

Set up control block areas.

Optimize.

 Process program options (default and specified), JCL information, and program control statements.

Optimize control field specifications.

Check Peerage technique criteria (listed in Section 1 under 'Program Organization') to determine whether this is a Peerage technique sort; if the criteria are not met, check Vale criteria (also listed in Section 1). 4. Initialize for Peerage or Vale Sort:

Generate code to handle control fields.

Make the program reenterable.

Set up control block areas.

Optimize.

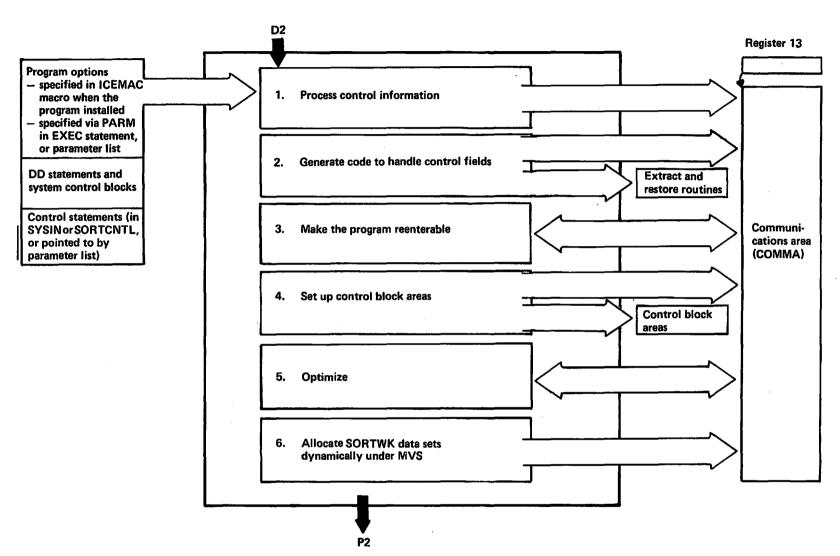
5. Initialize for <u>conventional Sort</u> or <u>Merge</u>:

Build CPI.

Prepare user routines.

Set up program control areas.

Optimize.



Extended Description for Diagram D3.1

Module	
DEF	3. Move code which is not reenterable to a reserved area:
DEF	- tree handling routines - special code 4. Create a control block area for each SORTWK data set. The area holds IOB, DCB, ECB, link fields, and other information.
	5. Determine: - no. and type of input buffers - internal blocksize
DEF	 index size maximum M for phases 2 and 3
DED	Optimize the use of the intermediate storage data sets. Optimize the mix of SORTIN/SORTOUT
DEC	buffers and SORTWK buffers, given the way in which the files are allocated, the data transfer rate of the devices, etc.
DEG	6. Allocate the number of specified (or defaulted) work areas with the space calculated or using the system's default value. Reallocate VIO data sets with a DSNAME specified, if required.
	DEF DEF DEC

Licensed Material--Property of IBM Page of LY33-8042-6, As Updated 31 March 1981, By TNL LN20-9345

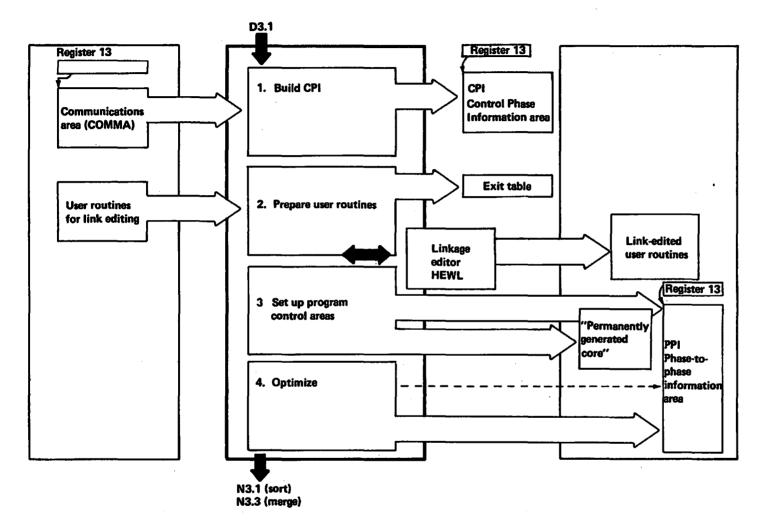
Module DEG

DEG

DEV

DEV

DEG



2

ensed Material--Property of IB

- 1. Move information from the communications area to the CPI. Release the communications area. If SIZE (MAX) is in operation, calculate how much virtual storage is available for sorting purposes, and note the amount in the CPI. The CPI is described in Section 5.
- 2. If user routines are present:
 - a. Copy any routines in SYSIN to SORTMODS.
 - b. If necessary, prepare linkage editor control statements, and put them in SYSLIN.
 - c. If necessary, create a parameter list for the linkage editor.
 - d. Build an exit name table for exits that do not require further linkage editing.
- 3. a. Transfer the information in the CPI to the PPI. The PPI is described in Section 5.
 - b. Get the storage necessary for the

- 'permanently generated core' area, to contain the exit name table, the format track table, and the control field handling routine.
- c. Generate the appropriate control field handling routine,

4. For a Sort

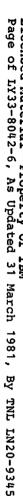
- a. Choose the sequence distribution technique to be used.
- b. Calculate bin size.
- c. Calculate variables:
 - B (sort blocking) and G (RSA capacity)
 - maximum M (defined in Section 3 under 'Phase 2')
 - the number of input and output buffers to be used.

For a Merge

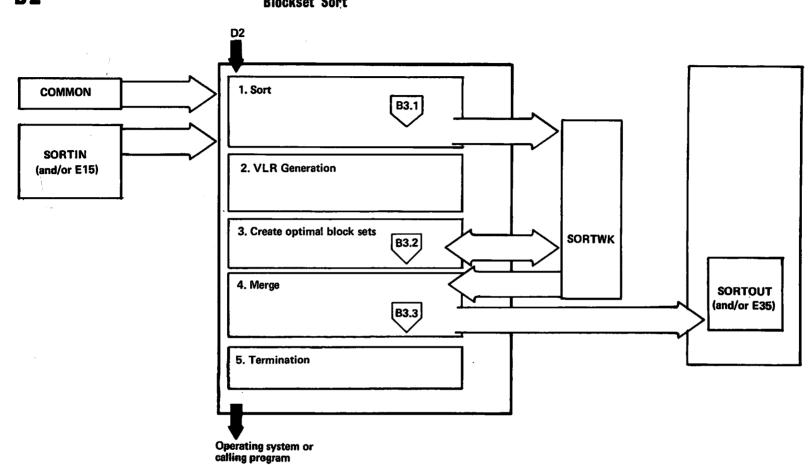
d. Determine the number of input and output buffers to be used.

Function	Conditions	Mod			
1 Create CPI	Always	RCM			
2a Copy rtns	Always	RCH			
b Link ed. stmts	Always	RCP			
c Parameter list	Always	RCM			
d Name table	Always	RCH			
3a Create PPI	Always	RC 1			
b Get storage	Always	RC 1			
c Gen multiple rtn	Always	AOL			
Gen extract (1)	Always	AOM			
Gen extract (2)	E61 activated*	A05			
	Otherwise	AO 4			
		RCR			
4a Choose technique	Tape	RCS			
	_	BGA			
	Disk	RCN			
b Calc bin size	(As for 4a)				
c Calc variables	Tape (As for 4a)				
	Disk BALN	RCK			
	CRCX	BGB			
d Calc no. of merge		1			
buffers	Always	RCL			
*And/or control fields of type AC,AQ,CLO,CSL,CST.					

This page intentionally left blank







icensed Material--Property of ibn

Extended Description for Diagram B2

 Read input via input buffers into the RSA (Record Storage Area).

FLR: If end of SORTIN is reached when few or no blocks have been written to SORTWK, bypass the key handling phase. Otherwise, write records to SORTWK to make room for new records to be read into the RSA. Index the blocks written to SORTWK.

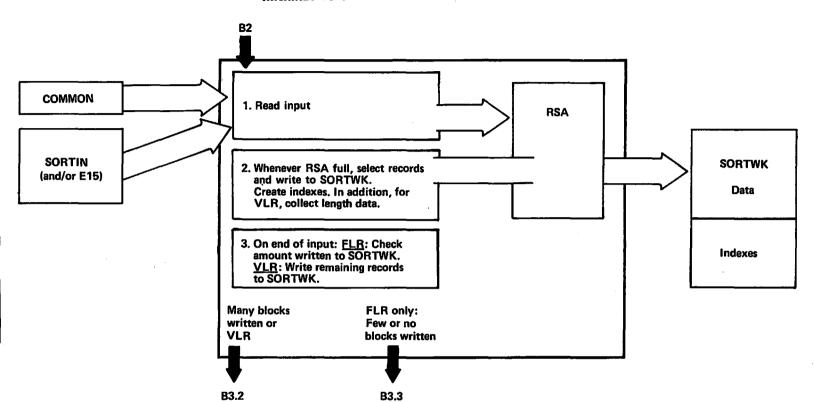
VLR: Write records to SORTWK to make room for new records to be read into the RSA. Collect record length data to use in computing bin size. Index the blocks written to SORTWK. At end of SORTIN, write remaining records to SORTWK and all index records.

 VLR Generation Phase: Compute bin size and generate code used by output phase to restore records.

- 3. Sort the indexes created by the SORT phase, and create optimal virtual block sets (physical blocks on same cylinder) for the merge phase.
- | 4. Merge virtual block sets to create the output data.
- 15. Terminate:

Print information messages. Close files used. Free storage used. 26





Extended Description for Diagram B3.1

1.	Get input	records	and	sto	ore,
1	partially	sorted,	in	the	Record
	Storage A	rea (RSA)	١.		

- a. From partially sorted RSA records select an output block continuing output string, or if no records for current output string begin a new one.
 - b. Select a cylinder on SORTWK data sets such that the keys in the

IPUM IPVM	
IPUM IPVM	
TPIIR	

IPVB

Module

block to be written are close to keys already written on cylinder.

- c. Write selected records to SORTWK, freeing RSA space for more input records.
- d. Make block indexes
- 3. <u>FLR:</u> If no SORTWK space or one string of data written to SORTWK, skip the key handling phase and go to Merge phase.

<u>VLR:</u> At input end of file, write remaining records and indexes to SORTWK. Call ICEGENV to initialize code for the OUTPUT phase.

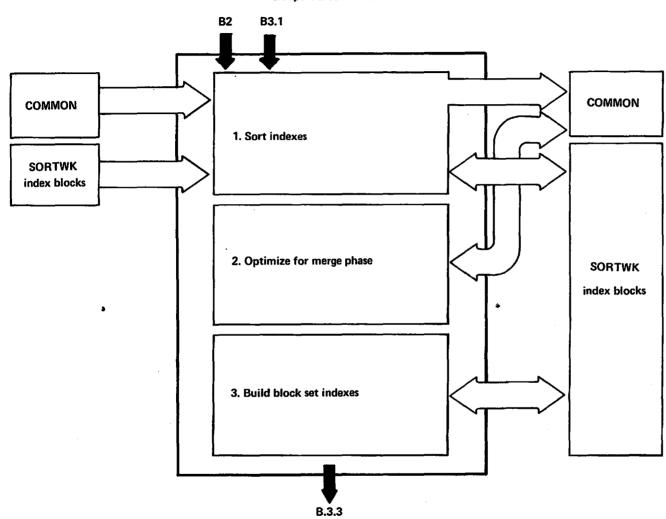
IPUB IPVB

IPUB

IPUT

IPVT

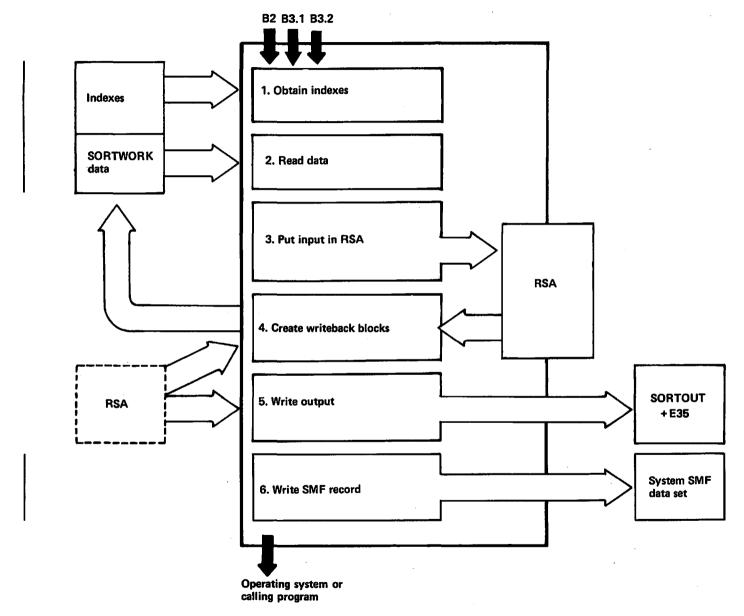
Licensed Material--Property of IBM of LY33-8042-6, As Updated 31 March 1981, By TNL LN20-9345



Module

Licensed Material -- Property of IBM As Updated 31 March 1981, By TNL LN20-9345

Module



Extended Description for Diagram B3.3

Select suitable writeback cylinder, and create indexes for writeback

blocks.

		Module]		
1.	Obtain (next) input virtual block index.	OPUT OPVT			If necessary spill indexes to SORTWK index blocks.
	If necessary read indexes from SORTWK, and/or build new virtual block indexes from writeback physical block indexes.		1	5.	Transfer records to output. If RSA and input buffers empty, finish processing. Otherwise return to point 3.
	If enough space in RSA for a buffer full of input, continue. Otherwise skip point 3.		1	6.	Build and write SMF record if requested (VS1, MVS only).
2.	Read data blocks.	OPUT			•
	Obtain disk addresses of virtual block set and read those in optimal order into input buffers.	OPVT			
3.	Unload input buffers into RSA. Return to point 1.	OPUA OPVA			
	If enough space can be created by moving RSA records to output, skip to point 5.	÷			•
4.	Select records from input buffers and RSA to form part of a writeback string, and write the string to SORTWK in the same way as during phase 1 (sort):	OPUT OPVT			

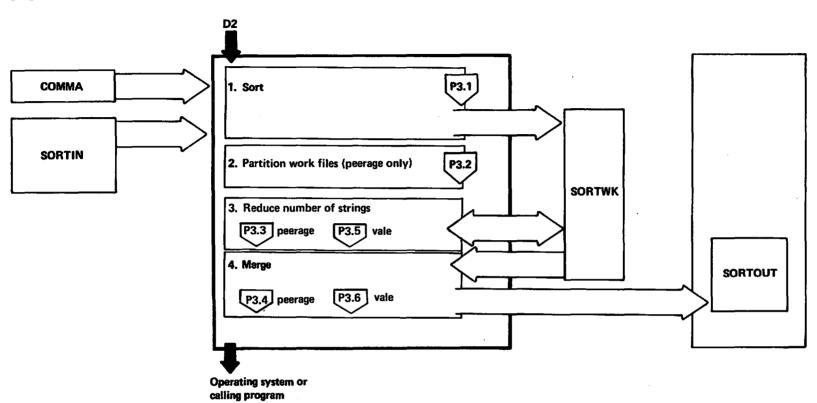
Licensed Material--Property of IBM Page of LY33-8042-6, As Updated 31 March 1981, By TNL LN20-9345

Module

OPUA

OPVA

OXOV



1. Read input via input buffers into the RSA (Record Storage Area).

If end of SORTIN is reached before the RSA is full, SORTOUT can be written immediately. The tree is used to select records for output.

Otherwise records are written out to SORTWK to make room for new records to be read into the RSA. For Peerage the blocks written to SORTWK are indexed.

Peerage

Sort the indexes, as described in MO diagram P4.21.

If the number of logical strings is <maximum M for phase 3, control is passed to the final merge (shown in MO diagram P3). M (merge order) is explained in Section 3 under 'Phase 1'.

If there are more strings, an intermediate merge pass is needed.

Partition the blocks on SORTWK, and arrange the indexes to reflect the partitioning, as described in MO diagrams P4.22 and P4.23.

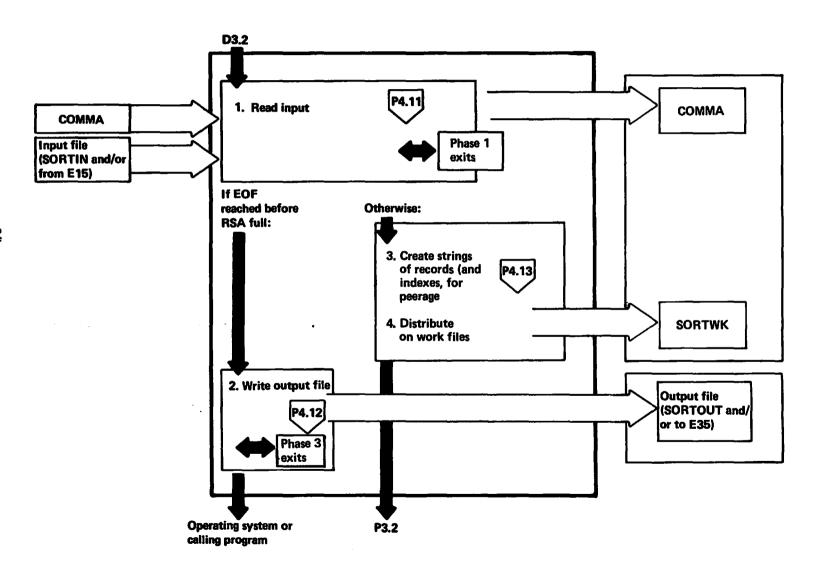
3. Merge all partitions except partition 0, creating new strings on SORTWK.

Return control to point 2.

4. All strings are now in partition 0.
Merge partition 0 to produce the output data set.

Vale

- 2. (Not used.)
- 3. Merge as many strings as possible (M for phase 2). Repeat until the number of strings is ≤ phase 3 merge order.
- 4. Merge the remaining strings to produce the output data set.



Move records into the RSA and update the tree for each record.

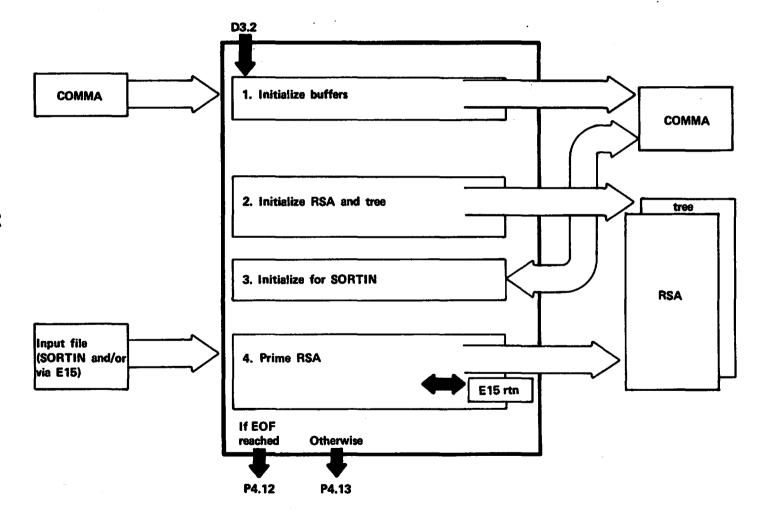
Continue until the RSA is full (or end of SORTIN is reached) .

2. If end of SORTIN is reached before the RSA is filled, write the output file.

3. Otherwise, intialize SORTWK data sets.

Create strings of records (and, for Peerage, indexes) for each output block.

4. Write the strings (and indexes) to SORTWK.



Licensed Material--Property of IBM

Extended Description for Diagram P4.11

 Allocate and initialize buffers for SORTWK data sets. Chain the buffer areas together. Allocate and initialize buffers used for SORTIN. (The same buffer area is used for SORTOUT if written in this phase.)

Allocate all remaining main storage to the RSA and tree. Chain bins in the RSA.

 Initialize addresses for GET and EOF routines. Load phase 1 routines if specified.

4. Get a record (from the input buffer or from E15).
Transpose the record.

Put the record in the next vacant bir in the RSA. Put the address of the record in the correct place in the tree.

Repeat until the RSA is full.

Module

CRE/VRE

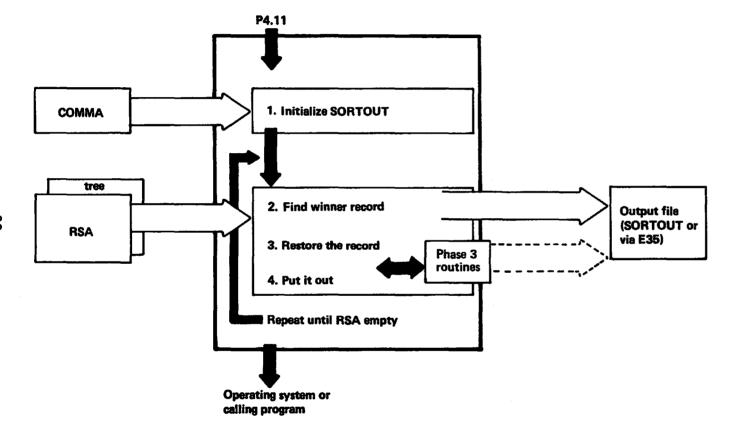
CRE/VRE

CRE/VRE*

DEG*

If EOF is reached before the RSA has been filled, the records do not need to be written to a SORTWK file. Instead, the SORTOUT file is written, using the tree to locate successive winner records in the RSA.

*The code is generated outside CRE/VRE's area; the source code is held in ICEDEG.



Extended Description for Diagram P4.12

1. a. Allocate and initialize buffers for SORTOUT. Initialize address for the appropriate put routine. Load E35 routine (if any).

b. Open SORTOUT if it is not a VSAM data set. Close the SORTIN data set.

2. Use the tree to locate the winner record in the RSA.

3. Restore the record to its original format. Move the record to the output buffer (or pass it to the E35 routine).

4. a. If VSAM, put the record to SORTOUT.

b. If non-VSAM, repeat 2 and 3a until the output buffer is full. Write the buffer contents.

Repeat points 2-4 until the RSA is empty.

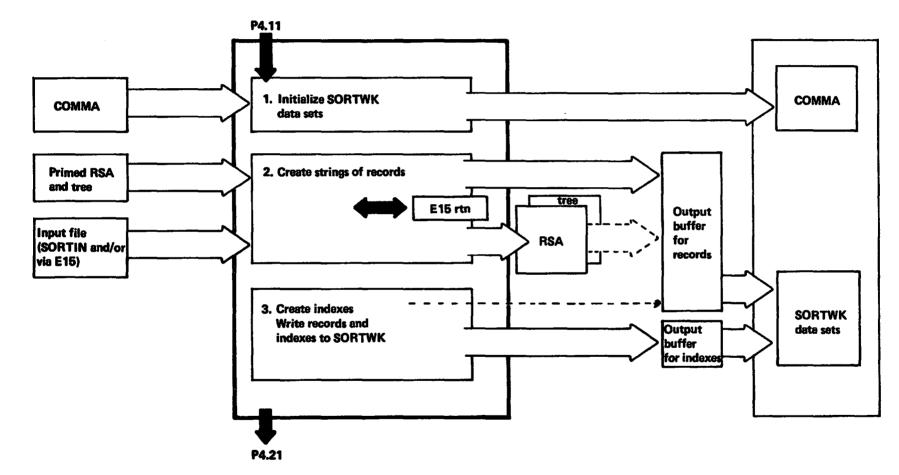
*The code which carries out these functions is generated outside ICECRO/VRO's area, and resides in ICEDEG. Module

CRO/VRO

*CRO/VRO

CRO/VRO

CRO/VRO/XCP



- 1. Set EOF and PUT routine addresses.
- Move the winner record from the RSA to the output buffer.
 Read the next record from the input buffer.
 Transpose the record.
 Move the record to the bin in the RSA from which the previous winner was moved.
 Update the tree.
- 3. a. Not used for Vale. Create a start index for the logical block, using the control word from the record in the output buffer. Put the index in the output buffer for indexes.
 - b. Repeat point 2 until the output buffer is full.
 - c. Write the block to SORTWK.
 - d. Not used for Vale. Create an end index for the logical block, using the control word from the last record moved to the output buffer. Put the index in the output buffer for indexes.
 - e. Not used for Vale. Repeat points 2 and 3a-e until the end of a string is reached. At end of string, make up a complete logical block, i.e. fill the last track with dummy physical blocks, so that the next string will begin a new track.

Repeat points 2 and 3 until EOF is reached.

Module Peerage	Vale
CRE	VRE/VRN
DEG	DEG
CRE	
CRE	VRE/VRN
CRE	
CRE	

Peerage only (not used for Vale) .

- Read the indexes. Sort them into strings. Index the strings, in the same way as record strings are indexed. Write the indexes to SORTWK. If necessary, merge the higher levels of indexes, until the original indexes are all in one string on SORTWK.
- 2. In calculating the number of logical strings, physical strings are ignored: only logical blocks are considered. The basis of the algorithm used is that two physical blocks whose contents overlap must necessarily belong to different logical strings. Two blocks whose contents do not overlap, however, can be included in the same string. The indexes can be used to determine whether or not blocks overlap, since they contain the control words of the highest and lowest records contained in the block.
- 3. Assign each logical block of records to a partition (partitions 0 to n), as described in P4.22. Put the assigned partition number into each index. Partitioning is a form of priming for the following merge step. It enables the merge to read in blocks in the most efficient order, to keep the number of merge passes to a minimum.
- 4. Sort the indexes (as in point 1), using the partition number as the first control field.

Module PAR PAR PAR

Peerage	only	(not	used	for	Vale)
---------	------	------	------	-----	-------

1. Read in the indexes.

2. Sort the indexes, in the same way as records are sorted in phase 1 (see MO diagram P4.12), using an RSA and tree. The first four fields of the index are used as control fields; the partition number in every index is always zero.

3. For each logical block of indexes written out, create a new, second-level index and write it. If the second-level indexes cannot all be contained in one string, create a third level, and so on, until a level is reached at which all indexes can be contained in one string.

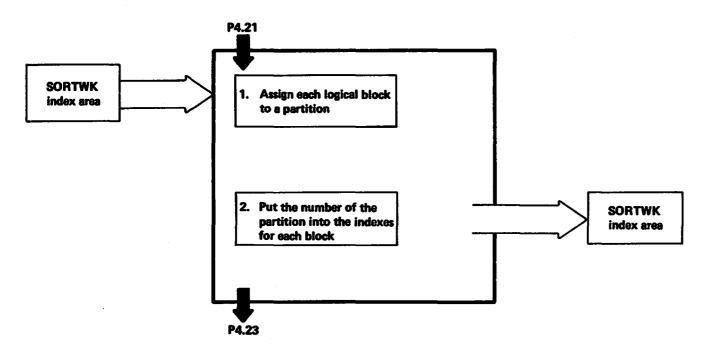
If all the sorted first-level indexes are contained in one string, skip points 4 and 5.

- 4. Carry out successive merges, until only two levels of index remain. Produce a single string from all the first-level indexes.
- 5. Write the string to SORTWK.

Discard second and higher-level indexes.

Module PAR PAR PAR PAR PAR PAR PAR

Partition Blocks



Extended Description for Diagram P4.22

Peerage only (not used for Vale)

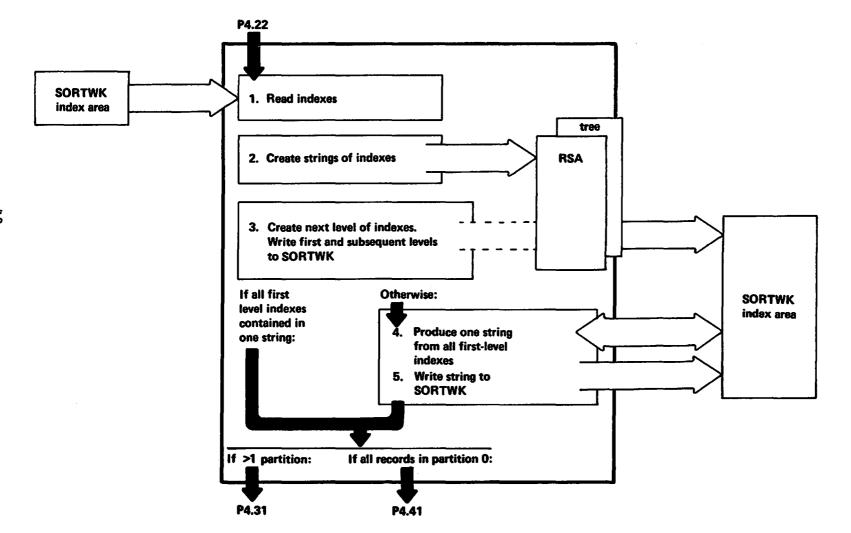
- a Read in the string of indexes created in P4.21.
 To reduce data handling, the indexes are 'mapped': an 8-byte map entry is created for each pair of indexes, using a hash table.
 - b Sort the map entries into partitions, in such a way as to maximize the number of logical blocks contained in each successive partition.

 Partition 0 contains M-1 strings, where M is maximum M for phase 3.

 Each of the other partitions contains M strings, where M is maximum M for phase 2. The last partition contains however many strings remain.
- Using the hash table, transfer the assigned partition number from each map entry to the corresponding pair of indexes.

Module PAR PAR PAR PAR

the second second second



£

Extended Description for Diagram P4.23

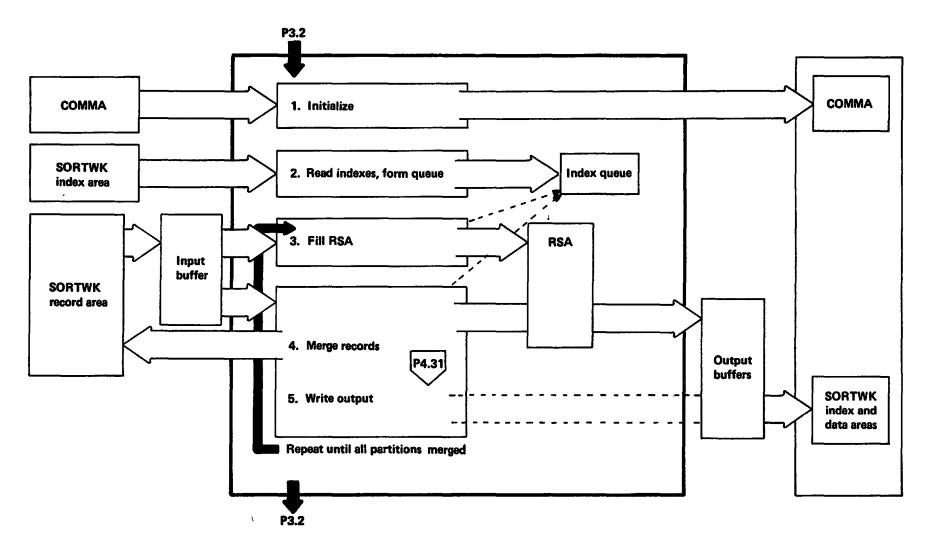
Peerage only (not used by Vale) .

The functions are identical to those performed in p4.21, and are carried out by the same segments.

The index sorting carried out here differs from that described in p4.21 in two respects:

- Indexes entering the routine have been updated to contain the correct partition number.
- Entry is from the partitioning routine, and exit is to a merge routine (intermediate or final). In p4.21, entry is from the sort or intermediate merge routine, and exit is to the partitioning routine.

Intermediate Merge (Peerage)



Extended Description for Diagram P3.3

An intermediate merge merges strings passed from the partitioning routine into a smaller number of longer strings on SORTWK.

It makes one pass for each partition except partition 0, producing one string from each, and then returns control to the partitioning routine. The partitioning routine continues to call the intermediate merge until the strings can all be contained in partition 0, when it hands control to phase 3 for the final merge.

- a. Allocate and initialize for SORTWK data sets.
 - b. Allocate all remaining main storage to RSA. Chain bins together.
- 2. a. Read in a block of indexes.
 - b. Queue the indexes. Each time an index is removed from the queue, bring in another and merge it into the queue.

RED RED RED RED

Module

- 3. Read a logical block of records from the address given in the first index in the queue. Merge the records into the RSA. Remove the index from the queue.

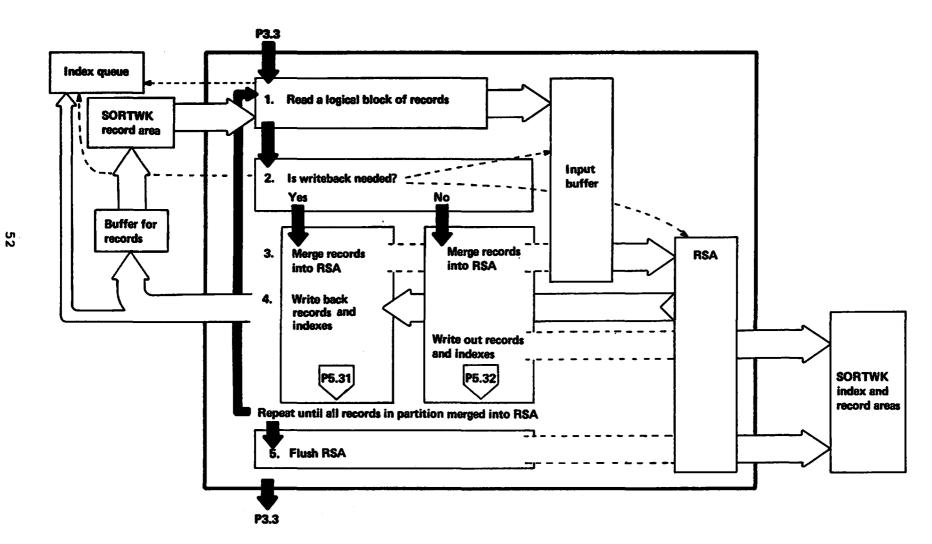
 Repeat until the RSA is full.
- 4,5 Read a block of records from SORTWK to an input buffer.

 If necessary, read a block back from the RSA to make room for the incoming block (the criterion for writeback is described in MO diagram P4.31).

 Otherwise write a block out from the RSA, and create indexes for it.

 Merge the incoming block into the RSA. Continue until the entire partition has been merged and indexed into one string on SORTWK.

Carry out points 3-5 for each partition except Partition 0.



Licensed Material--Property of IBM

Extended Description for Diagram P4.31

1.	Read :	into the	input	buffer	the	logical
	block	of reco	rds po:	inted (to by	the
	first	index i	n the :	index o	jue ue .	•

2. Find out whether enough records can be written out of the RSA to make room for the incoming block.

The space which can be made available is equal to the number of empty bins, plus the number of records (in the RSA and in the incoming block) which 'win' over the first record in the next block that will be read in (pointed to by the next index in the queue). If this space is smaller than the number of records in the incoming block, writeback is needed.

Merge the incoming block into the RSA, each record forcing out one to a buffer.

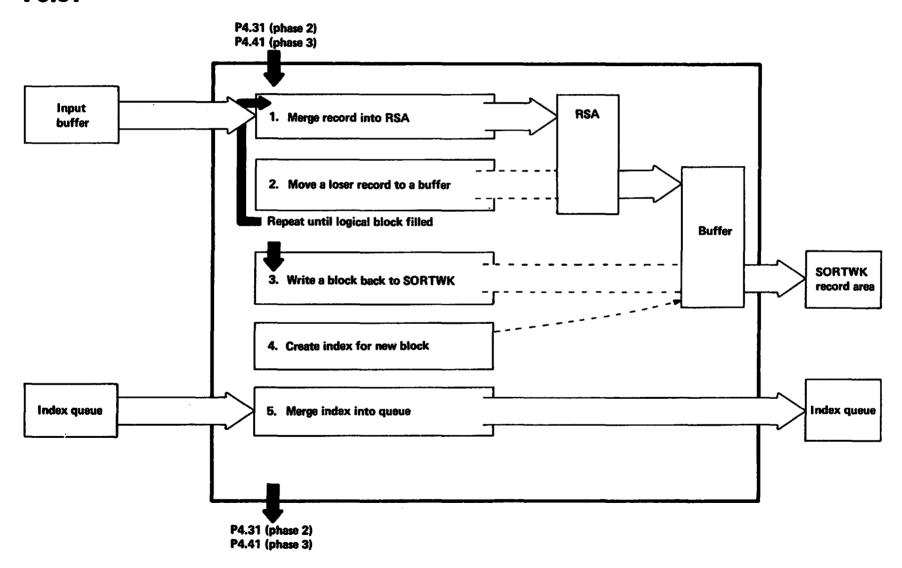
4. Create an index for the new block and move the block to SORTWK.

If writeback is required, add the index to the index queue; otherwise, write out the index to SORTWK.

5. When there is no more input in this partition, flush out the records remaining in the RSA. The partition is now in one string on SORTWK.

Module	
RED	
RED	
RED RED	
RED	
RED	
RED	

P5.31 Writeback



Extended Description for Diagram P5.31

- 1. Move the first record from the input buffer into the first vacant bin in the RSA, and chain into its correct position.
- 2. Move a record from the losing half of the RSA out to a buffer.
- When a logical block is complete, move it to SORTWK.
- 4. Create an index for the new logical block.
- 5. Merge the new index into the index queue (from which blocks of records will be selected for reading in to the current merge pass).

*** = RED in phase 2, LIM/LIV in phase 3.

Module *** *** ***

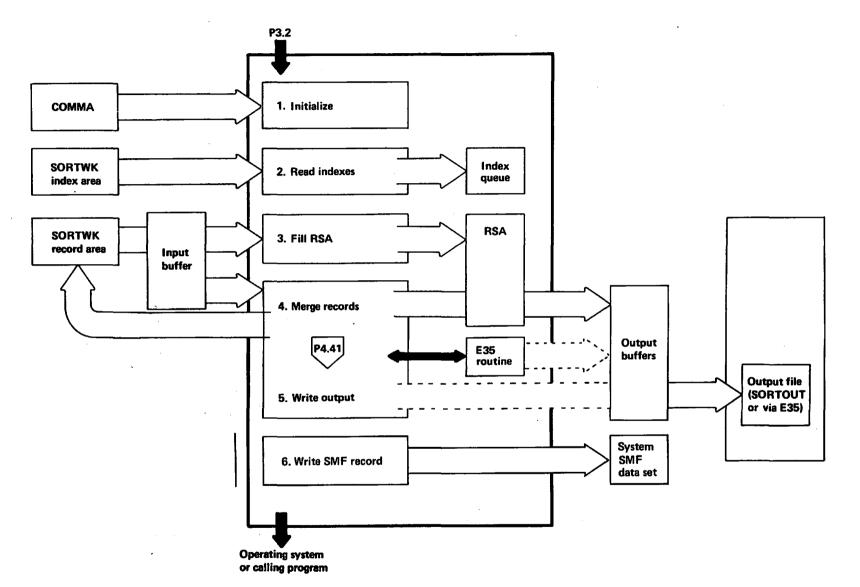
Extended Description for Diagram P5.32

The difference between this function and P5.31 (writeback) is in the handling of the indexes: here, they are written out to the area passed to the partitioning routine; in writeback, they are added to the index queue, so that the block to which they refer will be read in again in the current merge pass.

- Move the first record in the input buffer into the first vacant bin in the RSA. Chain it into its correct position.
- 2. Move the winner record from the RSA to the output buffer.
- When a logical block is complete, move it to SORTWK.
- Create indexes for the new logical block.
- 5. Move the new indexes to the index output buffer; when the buffer is full, write it to SORTWK.

Module RED RED RED RED RED

58



Extended Description for Diagram P3.4

The differences between the final merge and the intermediate merge (P3.3) are that: input consists of only one partition, partition 0; no new indexes are created; and output is to SORTOUT (or equivalent), not to SORTWK.

- 1. a. Allocate and initialize for SORTOUT and SORTWK data sets. Open the SORTOUT data set if it is not VSAM.
 - b. Allocate all remaining main storage to the RSA. Chain bins together.
- 2. a. Read in a block of indexes.
 - b. Queue the indexes. Each time an index is removed from the queue, bring in another and merge it into the queue.
- 3. Read a logical block of records from the address given in the first index in the queue. Merge the records into the RSA. Remove the index from the aueue. Repeat until the RSA is full.

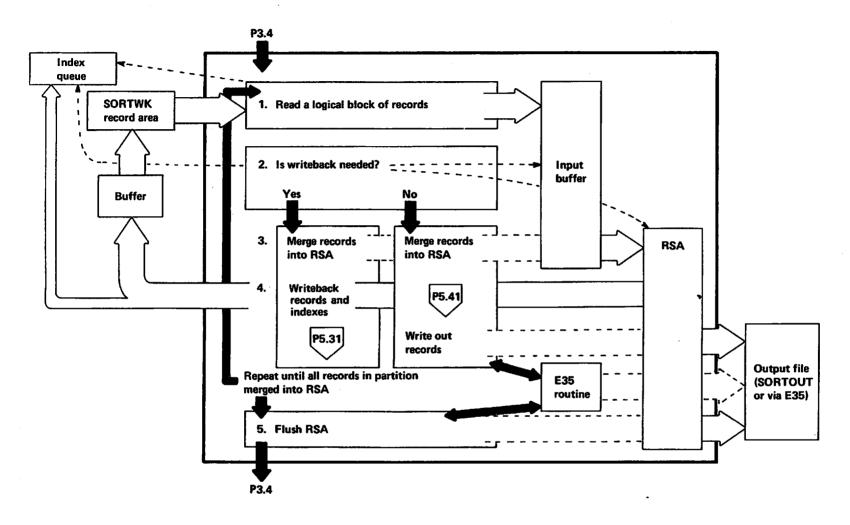
Module LIM/LIV LIM/LIV LIM/LIV LIM/LIV LIM/LIV

4.5 Read a block of records from SORTWK to an input buffer. If necessary, read a block back from the RSA to make room for the incoming block (the criterion for writeback is described in MO diagram P4.31). Otherwise write a block out from the RSA, taking E35 after each record is moved out. Merge the incoming block into the RSA. Continue until the entire partition has been merged into one string on SORTOUT.

6. Write SMF record type 16 to system SMF data set, if requested.

Licensed Material--Property of IBM As Updated 31 March 1981, By TNL LN20-9345

LY33-8042-6,



Extended Description for Diagram P4.41

l	1.	Read into the input buffer the logical
		block of records pointed to by the
		first index in the index queue.

2. Find out whether enough records can be written out of the RSA to make room for the incoming block.

The space which can be made available is equal to the number of empty bins, plus the number of records (in the RSA and in the incoming block) that 'win' over the first record in the next block which will be read in (pointed to by the next index in the queue). If this space is smaller than the number of records in the incoming block, writeback is needed.

3. Merge the incoming block into the RSA, each record forcing out one to a buffer. Take exit E35.

| 4. Move the block to SORTWK.

If writeback is required, Create an index for the new block, and add the index to the index queue.

5. When there is no more input, flush out the records remaining in the RSA.

The data set is now in one string on SORTOUT.

Module

LIM/LIV

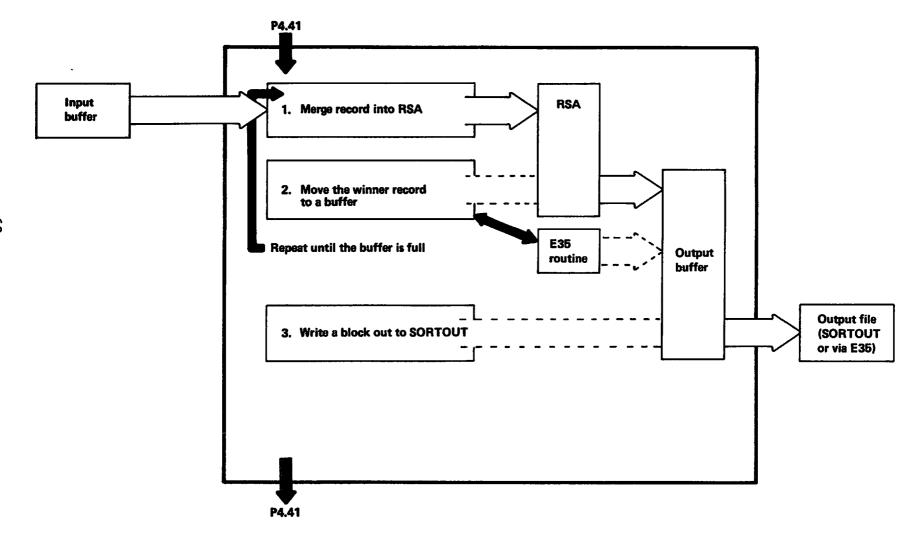
LIM/LIV

LIM/LIV* LIM/LIV

LIM/LIV

LIM/LIV/XCP

^{*}Code generated outside LIM's area; address kept in COMMA at CMERGE.



62

- 1. Move the first record in the input buffer into the first vacant bin in the RSA. Chain it into its correct position.
- 2. Take E35 if specified.
 Move the winner record from the RSA to
 the output buffer.
- 3. When the buffer is full, move it to SORTOUT.

These functions are carried out by module VED; if exit E25 has been specified an additional module, VEE, is used to handle the interface with the E25 routine.

- a. Allocate buffers: for input, phase
 2 merge order plus one; for output,
 two.
 - b. Initialize the phase work area.
 - c. Load and call E21 and load E25, if specified.
- 2. At the beginning of a merge pass:
 - a. Fill the input buffers.
 - b. Make a simple tree, with one pointer to the first record in each buffer, and the pointers (physically) in the desired output order.

Subsequently:

- c. Keep the pointers updated to point to the current record in each buffer.
- d. Keep the tree order so that the pointers are in the desired output order.
- e. Replenish the input buffers when

necessary, using the extra input buffer for 'smart lookahead'.

1 3. a. If E25 is not activated:

Move the record pointed to by the first pointer in the tree from its input buffer to the output buffer.

b. If E25 is activated:

Move the record pointed to by the first pointer in the tree from its input buffer to the E25 work area. Each new record moved to the work area forces the preceding record out to the output buffer. Take E25.

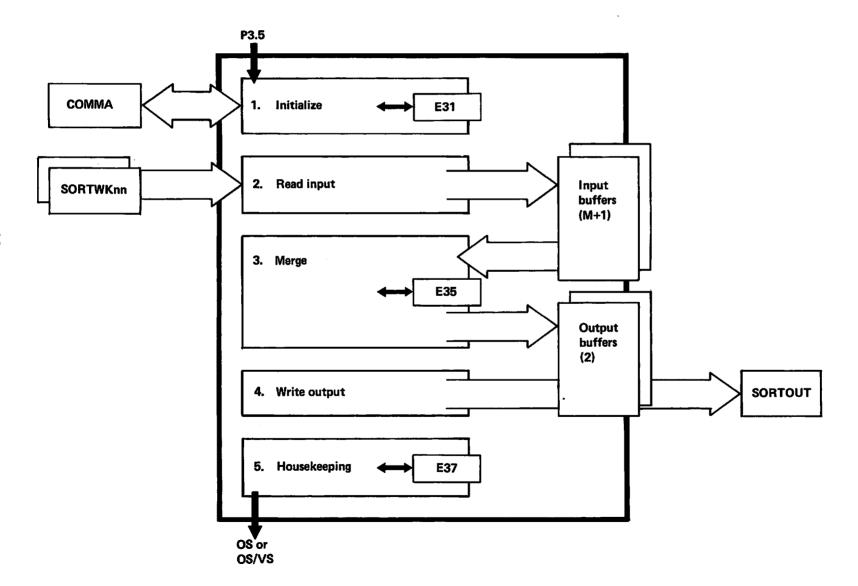
4. When the output buffer is full, write it to a SORTWK data set and chain it into the correct place in the string.

Repeat 2c-4 until input is exhausted. If the number of output strings is now less than or equal to maximum phase 3 M, continue to point 5. Otherwise, return to 2a for another merge pass.

- 5. a. Load and call E27 if specified.
- b. E25 housekeeping.
 - c. Free temporarily allocated areas.

P3.6

Final Merge (Vale)



67

Extended Description for Diagram P3.6

- 1. a. Allocate buffers: for input, phase
 3 M plus one; for output, the
 number determined in phase 0.
 - b. Initialize the phase work area.
 - c. Load and call E31 if specified.
- 2. At the beginning of the phase:
 - a. Fill the input buffers.
 - b. Make a simple tree, with one pointer to the first record in each buffer, and the pointers (physically) in the desired output order.

Subsequently:

- c. Keep the pointers updated to point to the current record in each input buffer.
- d. Keep the tree ordered so that the pointers are in the desired output order.
- e. Replenish the input buffers when necessary, using the extra input buffer for 'smart lookahead'.

- 3. If E35 has been specified:
 - a. Move the record pointed to by the first pointer in the tree from its input buffer to a work area, restoring its original format.
 - b. Call E35.
 - c. On return, move the record passed by E35 to an output buffer.

Otherwise, move the record pointed to by the first pointer in the tree from its input buffer to an output buffer, restoring its original format.

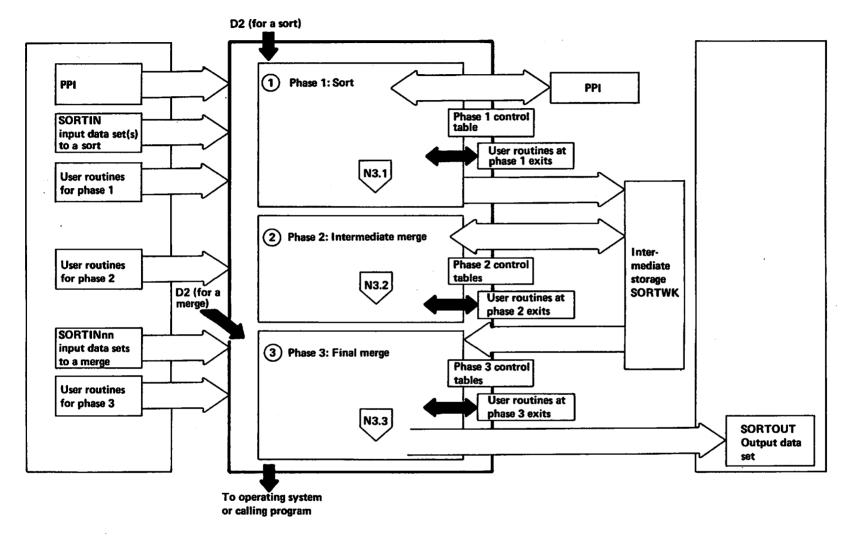
4. When the output buffer is full, write it to the SORTOUT data set.

Repeat 2c-4 until input is exhausted.

- 5. a. Load and call E37 if specified.
 - b. Delete all exit routines.
 - c. Free temporarily allocated areas.

Licensed Material--Property of IBM As Updated 31 March 1981, By TNL LN20-9345

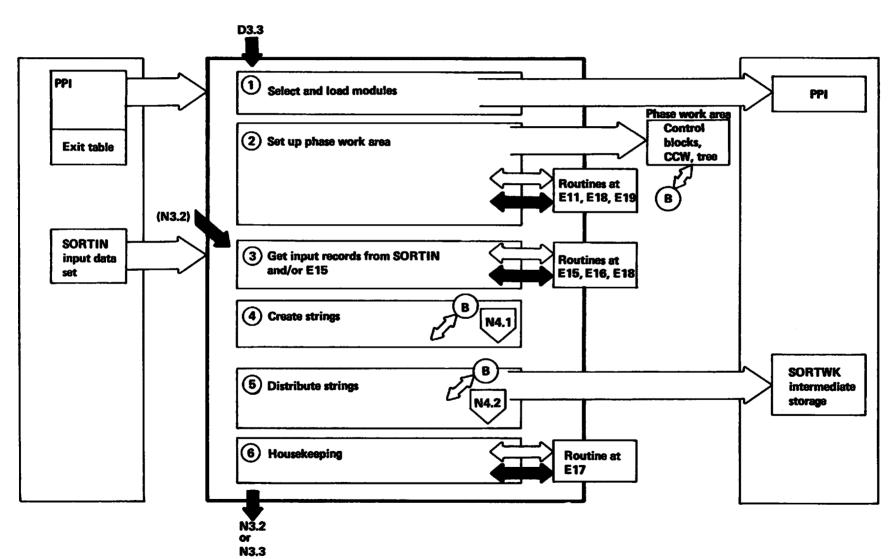
d. Write SMF record, if requested.



2. Initialize for phase 2. Take E21, E28, E29. Read strings of records from SORTWK. Take E28.

- Merge the strings into longer strings. Take E61, E25. Write out the new strings to SORTWK. Take E29, E27.
- 3. Initialize for phase 3. Take E31, E38 and E39. Read strings of records from SORTWK. Take E32, E38. Merge strings into output buffer. Take E61. Write out single string to SORTOUT. Take E35,E39,E37. Perform final housekeeping.

70



- 1. The selection of modules is made by ICERC6 using information from PPISW1 combined with the module table TBLPH1RN (which is defined in ICERC6).

 Information about user routines not link edited by the program is taken from the Exit Name Table (. PPIAPGC). ICERC9 a) loads all running modules and user routines, then b) loads, executes and deletes the assignment modules one by one.
- 2. Take exit 11. Acquire main storage (through GETMAIN) for the phase work area, buffers, and RSA. Initialize the tree in the phase work area. Generate the move routine. Generate control blocks (DCB, IOB, ECB) and CCWs in the phase work area. Take exits E18 and E19. Define the RSA bin structure, and enter the relevant information in the PPI. Open the SORTIN and SORTWK data sets. Records from SORTIN are skipped if the SKIPREC option is present. Code in the running modules is initialized by the respective assignment modules.

If the CRCX or OSCL technique is being used some further functions are performed here which otherwise belong to

phase 2; see MO diagram N3.2 for details.

- 3. Read SORTIN, using QSAM or VSAM.
 Locate mode is normally used, but
 move mode is used for spanned
 records. Each deblocked
 variable-length record has its length
 checked. VSAM variable-length records
 are supplied with a record descriptor
 word (4 bytes at the front of the
 record).
 Take exit E15 (record may be altered
 or deleted or a new record inserted).
 If calculated capacity is exceeded,
 take exit E16.
- 4. Fill the RSA, updating the tree for each record. Create strings by moving records from the RSA to a buffer, using the tree to select each record to be moved.
- 5. Write output, using the EXCP macro instruction.

 The sequence in which strings are written to the various SORTWK data sets is determined by the sequence distribution technique used.
- 6. If exact file size was specified, check the record counts. Take exit E17. Close SORTWK data sets. Free main storage and delete modules.

Module Table for Diagram N3.1

Fui	nction	Con	dition	s		Mod
1a	Select modules	osci	L,CRCX	pha:	se 1	RC 6
	b VSAM read (additio	Oth	er			RC6
1b	Load modules	Alw	ays			RC9
2a	Allocate storage	CRC:	X,OSCI			APL
L		Oth	erwise			APG
2b	Build blocks and	POL	Y,BALN	i (t)		AGA
	tables	OSC	AGN			
l		BALI	N (đ)			AGI
		CRC				9GN
3a	3a Deblock	Fix	Exit	CRCX,OSCL		RDR
		1		Other	RDD	
			NO E	<256	CRCX,OSCL	RDQ
					Other	RDC
					CRCX,OSCL	RDR
					Other	RDB
		Var	Exit	CRCX,OSCL		RDS
				Other		RDE
l			No E	CRCX	,OSCL	RDT
L		<u> </u>	L	Other		RDG
3b	VSAM read (addition					RGV
6	Housekeeping (For CRCX & OSCL, carried out in ph2)		er tha	n CRO	CX,OSCL	RPC

7

This page intentionally left blank

Licensed Material -- Property of IF

- 1. a. Move a record from the input buffer to the RSA.
- b. Update the tree.
 - a. Move the winner from the RSA to the output buffer.
 - b. Move in the next record from the input buffer.
 - c. Update the tree. Repeat until there are no more input records; unless the OSCL or CRCX technique is being used, in which case repeat until M-1 strings have been moved to the output buffer.
 - Move remaining records from the RSA to the output buffer.

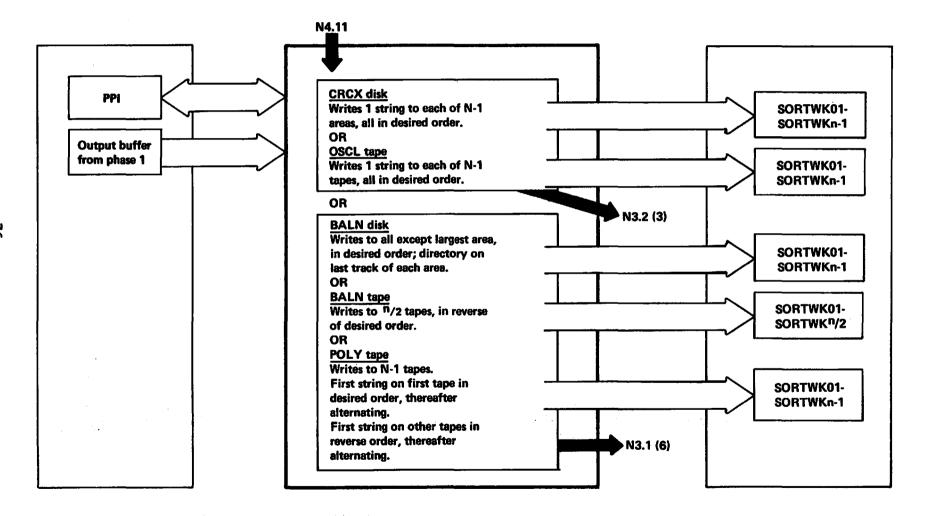
Module Table for Diagram N4.11

Function	Cond	ditions		Mod	
1a Move	Fix	Fix <256, no E			
		Other		ABS2	
	Var			RBF	
1b Update tree		Mult.	POLY	ROE	
-		I	Other	ROA	
	1	Single	POLY	ROF	
			Other	ROB	
	Var	Mult.	POLY	ROG	
			Other	ROC	
	1	Single	POLY	ROH	
	L		Other	ROD	
2a Block	Fix	256+	CRCX,OSCL	RBZ	
			Other	RBC	
		<256	CRCX,OSCL	RBY	
			Other	RBB	
	Var		CRCX,OSCL	RBA	
			Other	RBE	

No special move routine needed--the deblock routine used.

This module generates a move routine which is kept in main storage (pointed to by PPIBDSVA+4); if the EQUALS option is in operation, an additional routine is also generated (†PPIMOVEQ).

Distribute Strings



6

CRCX disk

One string is written to each of N-1 areas. If a physical area is too small to contain the whole string one or more tracks are borrowed from another physical area. These together make up the logical area. This can be done because each block within the string contains the disk address of the following block; the last block in a string contains its own address. Starting addresses of strings († KTABLEL) and free track list († KFRTRKLT) are kept in the phase work area.

BALN disk

Strings are written to N-1 areas in order of size, starting with the smallest. Eight strings are written to each area at a time, completely filling one directory block. If a complete string does not fit into the remainder of an area it is divided into substrings, the first completely filling the area.

The directory contains the starting address of the string, and the remaining blocks follow consecutively. The last block of the string contains a special EOS indicator.

BALN tape

Strings are written to N/2 tapes, one at a time. If a string does not fit on a tape it is divided into substrings.

OSCL_tape

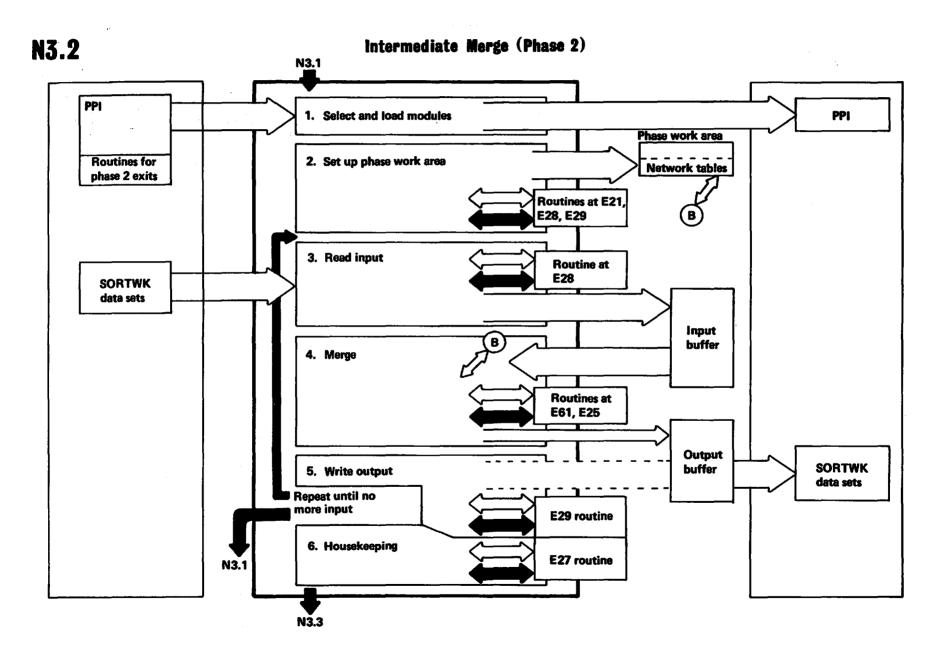
One string is written to each of N-1 tapes.

POLY tape

Strings are written to N-1 tapes, one at a time, until the lowest of the next level Fibonacci numbers is reached; then to N-2 tapes until the next number at the same level has been reached, and so on. When a complete level has been reached the next level numbers are calculated and the process is restarted. When all strings have been distributed dummy strings are added if necessary to make up Fibonacci numbers. The numbers of real and dummy strings are kept in PPITPTBL.

Module Table for Diagram N4.12

Function	Conditions	Mod
1a Sequence Distribu-	CRCX	80N
tion algorithm	OSCL	RON
_	BALN (disk)	ROK
	BALN (tape)	ROI
	POLY	ROJ
1b Write	CRCX	8PA
	BALN (disk)	· RPB
	Other	RPA



- The selection of modules is made by ICERC7 using information from PPISW1 and from the module table TBLPH2RN*. Exit information is taken from the Exit Name Table (†PPIAPGC). ICERC9 a) loads all running modules and user routines, then b) loads, executes and deletes the assignment modules sequentially.
- 2. Take exit E21*. Acquire main storage (through GETMAIN) for the phase work area and for buffers. Generate the record move routine. Build the input buffer table and network table in the phase work area*. Generate control blocks (DCB, IOB, ECB) and CCWs in the phase work area. Take exits E28 and E29*. Open the SORTWK data sets. Code in the running modules is initialized by the respective assignment modules.

Points 3, 4 and 5 are repeated until no input strings are left:

- 3. Read one string from each input tape--unless BALN (disk) is being used, in which case read M (or remaining, if less) strings at a time from one area (the largest of those with strings from the previous phase). For reading, the EXCP macro instruction is used.
- 4. Store the address of one record from each string (or its extracted control fields) in a network table (†PPINETAR). Take exit E61.

Find the next record to be moved to the output buffer, using binary search. Put the address of the next record in the "winner's" string in the network table. Take exit E25.

5. For writing, the EXCP macro instruction is used.
CRCX: The string is written to the area not used as input.
OSCL: The string is written to the next tape.
BALN (disk): The string is written to the output area left empty by the previous phase or emptied in this pass.
BALN (tape): The string is written to the next tape.
POLY: The string is written to the only available tape.

If the CRCX or OSCL technique is being used, the program returns from here to Phase 1 (see diagram N3.1), unless a) there is no more input, or b) the current level of strings now numbers M, in which case it repeats points 3-5.

6. Take exit E27. Check the record counts. Close the SORTWK data sets. ICERCT free main storage and deletes modules.

Unless the CRCX or OSCL technique is being used, the entire phase is re-executed until only one more merge pass is needed.

*When the CRCX or OSCL technique is used, these steps are executed in phase 1; see MO diagram N3.1

Module Table for diagram N3.2

Fu	nction	Con	ditio	ns		Mod
1a	Select modules	Mo	t CRC	Kor	OSCL)	RC7
	Load modules	Alw				RC9
	Allocate storage		t CRC	K or (OSCL)	APH
2b	Build control	BAL	N (dis	k)		AGJ
1	blocks and tables	POL	Y, BALI	N (tap	e)	AGG
3	Read		Y, BALI		forward	RGL
					backward	RGB
		OSC	L			RGB
1			N (d)			RGC
		CRC				8GB
4a	Compare, select,	Mul	tiple			ROP
	update tree	Sin	gle			ROQ
4b	Deblock/block	Fix	Exit			RBW
		1		Othe:		RBJ
			No E	256+	CRCX,OSCL	RBU
l			1		Other	RBH
			ļ.	<256	CRCX,OSCL	RBT
i					Other	RBG
l		Var	Exit			RBX
				Othe:		RBK
			No E	CRCX	OSCL_	RBV
			L	Othe		RBI
1	Move (additional t 256+ and/or with e	xits) ,	/bloc	k for fix	ABR*
5a	Sequence distribu-	CRC	X			80N
l	tion algorithm	OSC:				RON
	•		N (dis			ROT
		BAL	N (tape	₽)		ROR
		POL				ROS
5b	Write	CRC				8PA
1		OSC:				RPA
1			N (tape		LY	RPD
L			N (dis	k)		RPE
6	Housekeeping	CRC				8PM
		OSC				RPM
1		Oth	er			RPF

The move routine is generated by this module and kept in main storage (PPIBDSVA+8).

This page intentionally left blank

N3.3 Final Merge (Phase 3), or Merge Only Sort: from N3.2 or N3.1 Merge: from D3.3 PPI Exit table 1. Select and load modules PPI Phase 3 user routines Phase work area 2. Set up phase work area Routines at E31, E38, E39 SORTWK Routines at 3. Read input strings E32, E38 SORTINnn Routine at 4. Merge Output **buffer** SORTOUT 5. Write output data set Routines at E35, E39 Routine at 6. Housekeeping

To operating system or calling program

- 1. The selection of modules is made by ICERC8 using information from PPISW1 combined with the module table TBLPH3RN.

 Exit information is taken from the Exit Name Table (.PPIAPGC).

 ICERC9 first loads the running modules and user routines, then loads, executes and deletes the assignment modules sequentially.
- Take exit E31.
 Acquire main storage (through GETMAIN) for the phase work area and buffers.
 Generate the move routine in the phase work area.
 Build control blocks (DCB, IOB, ECB) and CCWs in the phase work area.
 Take exits E38 and E39.
 Open SORTOUT and the input data sets. Code in the running modules is initialized by the respective assignment modules.
- 3. For a sort, the strings are read from the SORTWK data sets.

 For a merge, the input strings are the SORTINnn data sets, or are supplied through E32.

- 4. Take exit E61.

 The address of one record from each input string (or of its extracted control fields) is kept in a network table (*PPINETAR). The next record to be moved to the output buffer is determined using binary search.

 The address of the next record in the 'winner's' string is put in the network table.
- 5. Take exit E35. The record may be altered or deleted or a new record inserted. The current record is then written to SORTOUT. The program uses QSAM or VSAM for writing output. Normally locate mode is used, but move mode is used for spanned records.
- 6. Take exit E37. Check record counts. Close data sets, free main storage, and delete modules.

Function	Con	ditio	ns		Mod
1a Select modules	Alwa	ays			RC8
1b Load modules	Alwa	ays			RC9
2a Allocate storage	Alwa				API
2b Build control		e sor			APK
blocks and tables	Dis	k sor	t		AGK
	Mer				APF
3a Read	POL	Y, Bali	(tape)		RGM
				backward	RGD
	OSCI				RGD
		(dis	k)		RGE
	CRC	~			8GC
	Mer	ge-on:	Ly	VSAM	RGV
21 2 2 2		1		OSAM	_ 4
3b Deblock (sort)	Fix	POLY	,BALN (t)		RDX
		017		back	RDH
	Var	Other	<u> </u>		RDH
Deblock (merge)		E32			RDI
peprock (merge)	No 1		spanne	5.5	RDU 8DJ
	NO I	.J.Z	other	<u> </u>	RDJ
4 Merge	Sing	rle	Tomer		ROQ
		iple			ROP
5a Block		Exit			RBM
		No E	256+		RBP
			<256		RBL
	Var	Exit	spanned	3	8BO
		<u> </u>	other		RBO
		No E	spanned	1	8BN
			other		RBN
5b Write (additional	to bl	lock f	or VSAI	M)(P	RPV
6 Housekeeping		-			RPG
¹Carried out by the a	ccess	meth	ođ.		

Section 3: Program Organization

Page of LY33-8042-6, As Updated 31 March 1981, By TNL LN20-9345

This section describes the structure of the sort/merge program, and of each of the phases that make up the program. For a sorting application all phases are normally used, as shown in Figure 1 in Section 1. For merging applications, only the first and last phases are used.

The interface between the various load modules is shown.

The phases are described, and a cross reference given to the appropriate MO diagrams in the Method of Operation section (Section 2). Record movement is also described.

Phase structures and storage layouts are shown in figures at the end of this section.

Program Exit Handling

Provisions are contained in Phases 1, 2 and 3 for inclusion of user routines at program exits. The linkage editor includes the user routines in the load module for the appropriate phase; however, when proper specifications are provided on the MODS control statements, link-editing does not occur, and the routines are loaded by the phase loader modules. See Appendix A for the various exits.

Module Interface

The interface of load modules in the program is shown in Figure 7.

While ICERCB is in main storage a patch area of 48 bytes (hex 30) is available. It is located at offset 1132 (hex 46C) from the start of the PPI area. Register 13 points to the PPI. The area is initialized with binary zeros.

When Phase 1 of Blockset (ICEIPUT for FLR, ICEIPVT for VLR) detects an error condition, it hands control to either ICEMESI or ICEMSGV, respectively. Phase 3 hands control to ICEMESO. When error conditions are detected in Phase 1, 2, or 3 of the other techniques, control is handed to the phase interface module (ICEMON for Peerage and Vale techniques, and ICERCV for the other techniques). The phase interface module handles the issuing of the appropriate message.

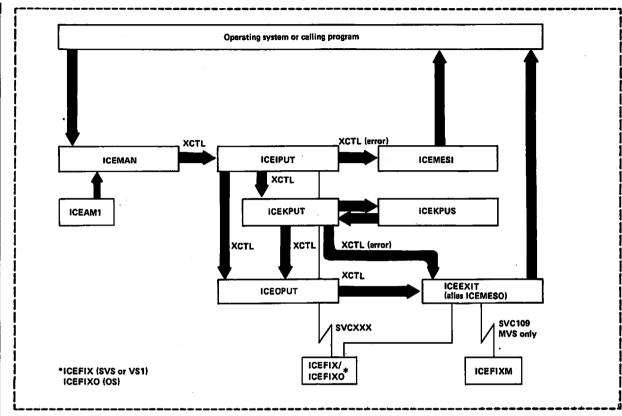


Figure 7. (1 of 4). Load Module Interface, FLR-Blockset

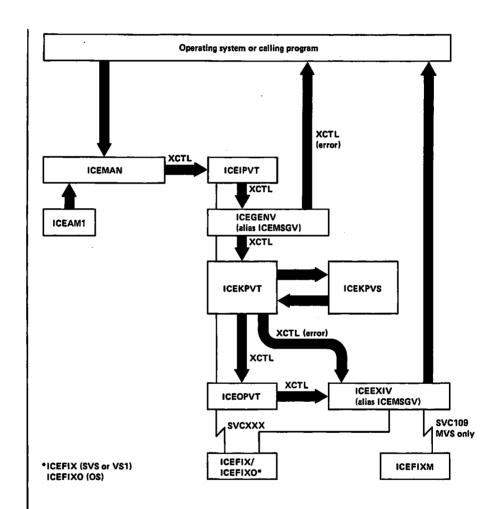
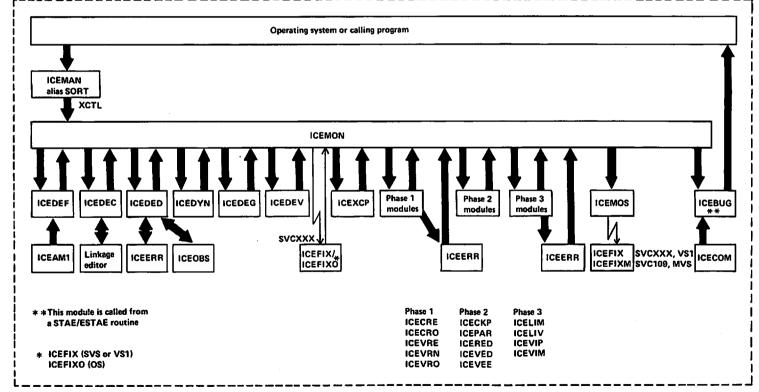


Figure 7. (2 of 4). Load Module Interface, VLR-Blockset



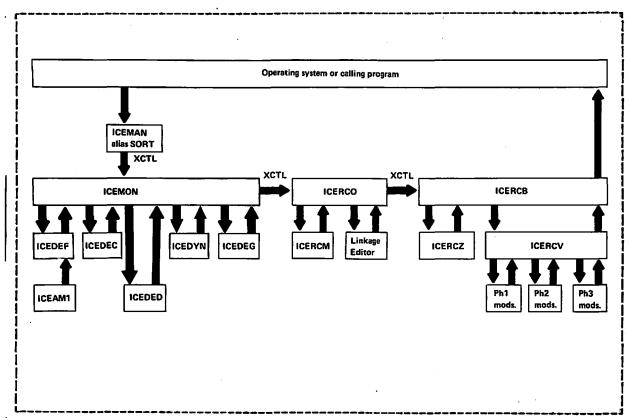
Page

of LY33-8042-6,

Licensed Material--Property of IBM As Updated 31 March 1981, By TNL LN20-9345

| Figure 7. (3 of 4). Load Module Interface, Peerage and Vale

Licensed Material--Property of IBM Page of LY33-8042-6, As Updated 31 March 1981, By TNL LN20-9345



| Figure 7. (4 of 4). Load Module Interface, Conventional Techniques

Licensed Material -- Property of IBM Page of LY33-8042-6, As Updated 31 March 1981, By TNL LN20-9345

Phase O

This phase collects the current system and sort control card information. It then decides which technique to be used and does the internal sort optimization. The functions are described in MO diagram D2. For conventional techniques, an SMF record type 16 is written to the system SMF data set, if requested.

Phase 1

Phase 1 performs the initial ordering of the input records. This one-pass phase (that is, each record is processed only once) that arranges the records of the input data set into ordered sequences. The sequences are written on the intermediate storage devices according to a predetermined distribution procedure.

With Blockset and Peerage, but not with Vale, an index entry is created for each work file block. This entry contains the control word of the first record in the block, and the block's disk address.

For Blockset applications this phase is described in MO diagram B3.1; for Peerage and Vale in MO diagram P3.1; for other techniques in MO diagram N3.1.

The input of records is controlled by EXCP, or by the SAM access method, and output is controlled by the sort/merge program -- unless the application is a standard disk sort which needs no work storage, in which case EXCP or SAM is used to produce the output data set. The sorting method used to order the records into sequences is a version of the replacement-selection technique.

When the conventional OSCL tape or CRCX disk techniques are used, control alternates between Phase 1 and Phase 2. With the BALN tape or disk technique, Phase 1 is first completed and then control is handed to either Phase 2 or Phase 3.

Phase 2

BLOCKSET TECHNIQUE

This phase processes the index entries from Phase 1 and assigns each block to a cylinder block set for merging. Two construction algorithms are used: one for a merge which writes only to the output file, and one for a merge which also writes to the work file.

This phase is described in MO diagram B3.2.

Licensed Material--Property of IBM

OTHER TECHNIQUES

For all other techniques, this phase combines the sequences produced by Phase 1 into a lesser number of longer sequences. It makes as many passes as necessary until the number of record sequences resulting from a given pass is equal to or less than the maximum Phase 3 merge order.

The merge order (M) is the number of sequences merged in any one merge pass. Maximum M for a given application is limited by the number of intermediate storage areas available, by the technique being used, and frequently by the number of buffers which can be accommodated in available main storage.

Phase 2 continues, therefore, until the total number of sequences remaining on all work areas is less than or equal to maximum M for the application in question.

Peerage Technique

This phase is described in MO diagrams P3.2 and P3.3.

The phase is in two parts, carried out by modules ICEPAR and ICERED. ICEPAR receives control after Phase 1 (ICECRE). It partitions the files, i.e. rearranges their indexes so that they will be read in by the merge in the most efficient order. If necessary, ICEPAR then initiates an intermediate merge: it instructs ICEMON to hand control to ICERED, which after merging returns to ICEPAR via ICEMON. This process continues until ICEPAR detects that the program is ready to enter Phase 3.

Vale Technique

This phase is described in MO diagram P3.5.

Each pass can merge up to 64 sequences. Two output buffers are used. The number of input buffers is M (merge order) + 1; the extra one is for 'smart lookahead', which means that when a buffer from a given sequence is emptied its next block should already be in main storage.

Conventional Techniques

This phase is described in MO diagram N3.2.

Each pass is capable of merging up to 16 previously sorted record sequences, or 63 if the balanced disk technique is used.

When the OSCL or CRCX techniques are used, control alternates between Phase 1 and Phase 2.

Phase 3

Phase 3 is a one-pass phase and produces a single ordered record sequence, thus completing the application.

For Peerage this phase is described in MO diagram P3.4; for Vale in P3.6; for Blockset in MO diagram B3.3; and for other techniques in N3.3.

In a sorting application, it follows Phase 2 (or Phase 1 if a Vale, Blockset, or conventional sort, skips Phase 2). In a merging application, it is executed immediately after Phase 0.

For a merging application, record input and output operations are performed by the control program's data management routines. For the final merge of a sorting application, Sort handles the record input operation, and EXCP or SAM is used to handle the record output operation, as in a merging case.

At the end of this phase for Blockset, Peerage, and Vale, an SMF record type 16 is generated and written to the system's SMF data set, if requested as an installation option.

Phase Structures

The Blockset, Peerage, and Vale technique phases consist normally of one load module each.

With other techniques the load modules are either planned overlay structures or dynamic structures. The organization of each phase with planned overlay or dynamic structures, and the modules that comprise them, are presented on the following pages. The modules for each phase are not necessarily placed in contiguous storage locations. In the case of dynamic phase structures, the modules are presented in the order in which they receive control.

Licensed Material--Property of IBM
Page of LY33-8042-6, As Updated 31 March 1981, By TNL LN20-9345

	MODULES, Peerage in 1.LINKLIB, or a private link library					
ICEAM1 ICEERR ICECKP ICELIM ICECRE ICELIV ICECRO ICEMAN* alias sort ICEDEC ICEMON ICEDED ICEMOS ICEDEF ICEOBS ICEDEG ICEPAR ICEDEV ICERED ICEDYN ICEXCP						
	INK MODULES, Peerage in a private link library.					
ICEBUG (used for producing a special formatted dump). ICECOM (source code for ICECOMMA.						
SVC MODULES, REENTERABLE, Peerage in SYS1.LPALIB or SYS1.SVCLIB.						
IGX00017: ICEFIX for SVS/VS1 ICEFIXM for MVS SVCXXX: ICEFIXO for MFT/MVT						
	n be changed by the user for SVS/VS1, med by the user for MFT/MVT.					
*See Blockset mo	dules for further information.					

Figure 8. Peerage Modules

REENTERABLE LOAD MODU SYS1.LPALIB, SYS1.LIN		vate link library				
ICEAMI ICECKP ICEDEC ICEDED ICEDEF ICEDEG ICEDEV ICEDYN ICEERR ICEMAN* alias so	ICEMON ICEMOS ICEOBS ICEVED ICEVEE ICEVIM ICEVIP ICEVRE ICEVRN	ICEVRO ICEXCP				
NONREENTERABLE LINK M SYS1.LINKLIB or a pri						
ICEBUG (used for producing a special formatted dump). ICECOM (source code for ICECOMMA.						
SVC MODULES, REENTERA						
IGX00017: ICEFIX for SVS/VS1 ICEFIXM for MVS SVCXXX: ICEFIXO for MFT/MVT						
The SVC names can be and must be renamed b						
*See Blockset modules	for further i	nformation.				

Figure 9. Vale Modules

Licensed Material--Property of IBM Page of LY33-8042-6, As Updated 31 March 1981, By TNL LN20-9345

Load Modules	Object Modules
ICEMAN alis SORT	ICEMAN ICEINIB ICEINIC ICEINID ICEINIO ICEDYNA ICEMESS ICEEXIN ICEFAUL ICEFAUL ICEMSGN ICEFORM ICEFVLN
ICEAM1	ICEAM1
ICEIPUT	ICEXPUT ICEE15B ICEFILI ICEIPUT ICEIPUM ICEIPUB ICECYLN ICEDEVT ICEEXII
ICEMESI	ICEMESI ICEMESS ICEEXII ICEMSGI ICESUBS
ICEKPUT	ICEKPUT ICEKPUV ICEKPUB ICEKPUB ICEMESS ICEEXIK ICEFILK ICECYLN ICEDEVT ICEMSGK ICEXPUT
ICEKPUS	ICEKPUS
SVC MODULES, REENTERABL SYS1.LPALIB or SYS1.SVC	E, FLR-Blockset in CLIB
IGX00017: ICEFIX for SUCCESIXM for N	17.5

Figure 10. (1 of 2). FLR-Blockset Modules

ICECOBU ICEE35B ICEFILO				
ICEOPUT ICEOPUA ICEOPUB ICELIMI ICECYLN ICEDEVT				
ICECKUT ICEDEVT ICEEXIO ICEEXIT ICEMESS ICEMSGO ICESUBS ICEXPUT				
SVC MODULES, REENTERABLE, FLR-Blockset in SYS1.LPALIB or SYS1.SVCLIB				
IGX00017: ICEFIX for SV5/VS1 ICEFIXM for MVS SVCXXX: ICEFIXO for MFT/MVT				

Figure 10. (2 of 2). FLR-Blockset Modules

Load Modules	Object Modules			
ICEMAN alis SORT	Same as FLR-BLOCKSET			
ICEAM1	ICEAM1			
ICEIPVT	ICEXPUT ICECOBV ICE15V ICEFVLI ICEIPVT ICEIPVM ICEIPVB ICECYLN ICEDEVT			
ICEGENV alias ICEMESV	ICEGENV ICEMESV ICEXII ICEXPUT ICESUBX			
ICEKPVT	ICEKPVT ICEKPVV ICEKPVA ICEKPVB ICEMESS ICEEXIK ICECYLH ICEMSGK ICEXPUT ICEDEVT ICEFVLK			
ICEKPVS	ICEKPVS			
ICEOPVT	ICE35VL ICEFVLO ICEOPVT ICEOPVA ICEOPVB ICECYLN ICEDEVT			
ICEEXIV alias ICEMSGV	ICEOXIV ICEOXOV ICEMESS ICEMSGO ICEXPUT ICESUBX ICEEXIV			
SVC MODULES, REENTERABLE, SYS1.LPALIB or SYS1.SVCLI				
IGX00017: ICEFIX for SVS/ ICEFIXM for MVS SVCXXX: ICEFIXD for MFT				

| Figure 10A. VLR-Blockset Modules

ICERCB ICERCM (overlay module, see Figure 12) **SORT* MODULES, NONREENTERABLE, Conventional Techniques in SYS1.SORTLIB or a private sort library **ICHABA ICHADL ICHAP2 ICHACQ ICHADH ICHACH ICHADH ICHACH ICHADH	NONKEEN: SYS1.LI	PERABLE INKLIB or	a priva	te link	library		s in	
ICEABA ICEADL ICEAP2 ICEACQ ICERDH ICEECH ICEGGN ICEABB ICEADP ICEAP3 ICEACG ICERDH ICEACG ICEABC ICEABC ICEABC ICEABC ICEACG ICEACG ICEACG ICEABC ICEABC ICEABC ICEACG ICEACG ICEACG ICEABC ICEABC ICEABC ICEACG ICEACG ICEACG ICEACG ICEABC ICEABC ICEACG IC			ICE ICE	RCM (ove	erlay mod	lule, se	e Figure e Figure	12) 12)
ICEABB ICEADP ICEAP3 ICEAGR ICERDI ICERGE ICEABC ICEABC ICEACK ICEAGS ICERDJ ICERGE ICEABC ICEABC ICEACK ICEACS ICERDJ ICERGE ICEABE ICEABR ICEADR ICEACK ICEACK ICEACC ICEACC ICEABC ICEADS ICEACK ICEACK ICEACC IC	'SORT' I	MODULES, RTLIB or	NONREEN a priva	TERABLE te sort	, Convent	tional T	echniques	in
ICEABC ICEADC ICECHK ICEAGS ICERDJ ICERCQ ICEABE ICEADE ICEADE ICEACT ICEACH ICEACH ICEACH ICEACS ICEACH ICEACH ICEACS ICEACH ICEACS IC	ICEABA	ICFADL		ICFACQ	ICERDH	ICEFON	ICE9GN	
ICEABE ICEADR ICEDM4 ICEACT ICEBEL ICERCR ICEABF ICEADS ICEX1 ICEACW ICERDP ICEACS ICEABG ICEADT ICEEX2 ICEACX ICEBDQ ICERCT ICEABH ICEADU ICEEX3 ICEBBH ICERCR ICEROW ICEABH ICEADX ICEBBA ICEBBH ICERCR ICEROX ICEABH ICEAGA ICEBBA ICEBH ICERDS ICEROX ICEABH ICEAGA ICEBB ICEBH ICERDT ICEROY ICEABH ICEAGA ICEBB ICEBH ICEBU ICEROZ ICEABH ICEAGC ICEBE ICEBH ICEBU ICERDZ ICEABH ICEAGC ICEBE ICEBH ICEBU ICERDA ICEABH ICEAGC ICEBH ICEBH ICEBU ICERDA ICEABH ICEAGC ICEAGH ICEBH ICEBH ICEBH ICEBH ICEABH ICEAGG ICEAGH ICEBH ICEBH ICEBH ICEBH ICEABH ICEAGG ICEAGH ICEBH ICEBH ICEBH ICEBH ICEABC ICEAGH ICEAGM ICEBH ICEBH ICEBH ICEABC ICEAGH ICEAGM ICEBH ICEBH ICEBH ICEABC ICEAGH ICEAMA ICEBH ICEBH ICEBH ICEABC ICEAGI ICEAMA ICEBH ICEBH ICEBH ICEABU ICEAOZ ICEAMA ICEBH ICEBH ICEBH ICEABU ICEAOZ ICEAMC ICEBH ICEBH ICEBH ICEABU ICEAPA ICEAOA ICEBH ICEBH ICEBH ICEABU ICEAPA ICEAOA ICEBH ICEBH ICEBC ICEABU ICEAPB ICEAOB ICEBZ ICEFMC ICEBCR ICEABU ICEAPB ICEAOC ICEBC ICEBC ICEBCR ICEABU ICEAPB ICEAOC ICERCT ICEGOA ICEBCR ICEADD ICEAPB ICEAOC ICERCT ICEGOA ICEBCR ICEADC ICEAPG ICEAOC ICERCT ICEGOB ICEBCR ICEADD ICEAPB ICEAOC ICEACC ICEGOB ICEBCR ICEADD ICEAPB ICEAOL ICEBCB ICERCE ICEGOB ICEADD ICEAPB ICEAOL ICEBCB ICERCE ICEGOB ICEADH ICEAOL ICEACI ICEGO ICEBCR ICEADH ICEAOL ICEACH ICEGOB ICEBCH ICEGOB ICEACH ICEACH ICEACH ICEACH ICEACH ICEACH ICEACH IC	ICEABB	ICEADP	ICEAP3	ICEACR	ICFRDI	ICERCP	ICE90N	
ICEABF ICEADS ICEEX1 ICEACW ICERDP ICERCS ICEABG ICEADT ICEEX2 ICEACX ICEEDQ ICERCT ICEABH ICEADU ICEEX3 ICEEBH ICERDR ICEROW ICEABH ICEADX ICEEBH ICERDS ICEROW ICEABH ICEADX ICEEBH ICEEBH ICERDS ICEROX ICEABH ICEAGA ICEEBH ICEEBH ICEEDT ICEROX ICEABH ICEAGA ICEEBH ICEEBH ICEEDU ICEROZ ICEABH ICEAGC ICEBBE ICEEBH ICEEDX ICEBPA ICEABH ICEAGC ICEBBE ICEEBH ICEEBH ICEEPH ICEABH ICEAGC ICEBBE ICEEBH ICEEBH ICEEBH ICEABH ICEAGC ICEAGK ICEEBH ICEEBH ICEEBH ICEABH ICEAGG ICEAGK ICEEBH ICEEBH ICEEBH ICEABH ICEAGG ICEAGK ICEEBH ICEEBH ICEEBH ICEABH ICEAGG ICEAGK ICEEBH ICEEBH ICEEBH ICEABH ICEAGH ICEAGH ICEEBH ICEEBH ICEEBH ICEABH ICEAGH ICEAGH ICEEBH ICEEBH ICEEBH ICEABH ICEAGI ICEAHA ICEEBH ICEEBH ICEEBH ICEABH ICEAOY ICEAHA ICEEBH ICEEBH ICEEBH ICEABH ICEAOY ICEAHA ICEEBH ICEEBH ICEABH ICEAOY ICEAHA ICEEBH ICEEBH ICEABH ICEADH ICEAOA ICEEBH ICEEBH ICEABH ICEADH ICEAOA ICEEBH ICEEBH ICEABH ICEAPH ICEAOA ICEEBH ICEEBH ICEABH ICEAPH ICEAOC ICERCT ICEEOA ICEEBH ICEABH ICEAPH ICEAOC ICERCT ICEECH ICEEBH ICEADH ICEAPH ICEAOH ICEEBH ICEECH ICEEBH ICEADH ICEAPH ICEAOH ICEACH ICEEBH ICEADH ICEAPH ICEAOH ICEACH ICEEBH ICEADH ICEACH ICEACH ICEEBH ICEADH ICEACH ICEACH ICEACH ICEEBH ICEACH ICEACH ICEACH ICEACH ICEEBH ICEACH ICEACH ICEACH ICEACH ICEACH ICEACH ICEACH ICEACH IC	ICFABC	ICFADÇ	ICECHK	ICEACS	ICERDJ	ICERCO	ICE9PA	
ICEABF ICEADS ICEEX1 ICEACW ICERDP ICERCS ICEABG ICEADT ICEEX2 ICEACX ICEEDQ ICERCT ICEABH ICEADU ICEEX3 ICEEBH ICERDR ICEROW ICEABH ICEADX ICEEBH ICERDR ICEROW ICEABH ICEADX ICEEBH ICEEBH ICERDS ICEROX ICEABH ICEAGA ICEEBH ICEEBH ICEEDT ICEROX ICEABH ICEAGA ICEEBH ICEEBH ICEEDU ICEROZ ICEABH ICEAGC ICEBBE ICEEBH ICEEDX ICERPA ICEABH ICEAGC ICEBBE ICEEBH ICEEBH ICEEPH ICEABH ICEAGC ICEBBE ICEEBH ICEEBH ICEEBH ICEEPH ICEABH ICEAGC ICEAGK ICEEBH ICEEBH ICEEPH ICEABH ICEAGG ICEAGK ICEEBH ICEEBH ICEEPH ICEABH ICEAGG ICEAGK ICEEBH ICEEBH ICEEPH ICEABH ICEAGG ICEAGK ICEEBH ICEEBH ICEEPH ICEABH ICEAGH ICEAGH ICEEBH ICEEBH ICEEPH ICEABH ICEAGH ICEAGH ICEEBH ICEEBH ICEEPH ICEABH ICEAGI ICEAHH ICEEBH ICEEBH ICEEPH ICEABH ICEAOY ICEAHH ICEEBH ICEEBH ICEEPH ICEABH ICEAOY ICEAHH ICEEBH ICEEBH ICEEBH ICEABH ICEAOY ICEAH ICEEBH ICEEBH ICEEBH ICEABH ICEAOY ICEAH ICEEBH ICEEBH ICEEBH ICEABH ICEAPH ICEAOO ICEEBY ICEEHH ICEBC ICEABH ICEAPH ICEAOO ICEERCY ICEEHH ICEBC ICEABH ICEAPH ICEAOC ICERCT ICEEOO ICEBCH ICEABH ICEAPH ICEAOC ICEECT ICEECH ICEBCH ICEADH ICEAPH ICEAOC ICEECT ICEECH ICEBCH ICEADH ICEAPH ICEAOH ICEECH ICEBCH ICEBCH ICEADH ICEAPH ICEAOH ICEBCH ICEBCH ICEBCH ICEADH ICEACH ICEACH ICEBCH ICEBCH ICEBCH ICEADH ICEACH ICEACH ICEBCH ICEBCH ICEBCH ICEACH ICEACH ICEACH ICEBCH ICEACH ICEACH ICEACH ICEACH ICEBCH ICEACH ICEAC			ICFDM4	ICFACT	ICEBEL	ICERCR		
ICEABH ICEADU ICEEX3 ICEBBH ICERDR ICEROW ICEABI ICEADX ICEBBA ICEBBI ICERDS ICEROX ICEABJ ICEAGA ICEBBB ICEBBJ ICERDT ICEBOY ICEABB ICEAGA ICEBBB ICEBBJ ICEBDT ICEBOY ICEABK ICEAGE ICEBBE ICEBBK ICEBDU ICEROZ ICEABL ICEAGC ICEBBE ICEBBL ICEEDX ICEBPA ICEABN ICEAGC ICEBBE ICEBBL ICEBDX ICEBPA ICEABN ICEAGC ICEBBE ICEBB ICEBBL ICEBBP ICEABN ICEAGC ICEBBE ICEBBC ICEBG ICEBPC ICEABN ICEAGC ICEAGK ICEBBC ICEBC ICEBPC ICEABN ICEAGG ICEAGL ICEBBP ICEBGE ICEBPC ICEABN ICEAGG ICEAGL ICEBBP ICEBGE ICEBPE ICEABR ICEAGG ICEAGN ICEBBP ICEBG ICEBPF ICEABR ICEAGI ICEANN ICEBBU ICEBG ICERPF ICEABS ICEAGI ICEANN ICEBBU ICERG ICERPF ICEABS ICEAGI ICEANN ICEBBU ICERG ICERPF ICEABS ICEANY ICEANN ICEBBU ICERCU ICEBPP ICEABU ICEAOZ ICEANC ICEBBX ICEBNA ICEBBN ICEABN ICEAPA ICEAON ICEBBY ICEFMC ICEBCR ICEABN ICEAPB ICEAOC ICERCT ICEBON ICEBCR ICEABN ICEAPC ICEAOC ICERCT ICEBON ICEBCB ICEADS ICEAPC ICEAOC ICERCT ICEBON ICEBGB ICEADS ICEAPF ICEAOC ICERCO ICERCO ICEBCO ICEADD ICEAPF ICEAOC ICERCO ICEBCO ICEBCO ICEADD ICEAPF ICEAOC ICERCO ICEBCO ICEBCO ICEADD ICEAPH ICEAOH ICEBCO ICEBCO ICEBPA ICEADC ICEAPG ICEAOG ICEBC ICEBCO ICEBPA ICEADC ICEAPG ICEAOG ICEBC ICEBCO ICEBPA ICEADC ICEAPG ICEAOG ICEBC ICEBCO ICEBCO ICEADD ICEAPT ICEAOI ICEBCO ICEBCO ICEBCO ICEBCO ICEADD ICEAPT ICEAOI ICEBCO ICEBCO ICEBCO ICEBCO ICEADD ICEAPT ICEAOI ICEBCO ICEBC	ICEABF	ICFADS	ICFEX1	ICEACW	ICERDP	ICERCS		
ICEABL ICEADX ICERBA ICERBI ICERDS ICEROX ICEABJ ICEAGA ICERBB ICEREJ ICERDT ICEROY ICEABK ICEAGB ICERBC ICEEBK ICERDU ICEROZ ICEABK ICEAGC ICERBE ICEEBL ICEEDX ICERPA ICEABM ICEAGC ICERBE ICEEBL ICEEBC ICERPA ICEABM ICEAGC ICERBE ICEEBC ICERGB ICERPC ICEABM ICEAGC ICEAGK ICEAGC ICERCD ICEABO ICEAGC ICEAGK ICEAGC ICERCD ICEABO ICEAGC ICEAGC ICERBC ICERPC ICEABC ICEAGC ICEAGC ICERBC ICERPC ICEABC ICEAGC ICEAGC ICERBC ICERPC ICEABC ICEAGC ICEAGC ICERCU ICERCF ICEABC ICEAGC ICEAGC ICERCU ICERCF ICEABS ICEAGC ICEACM ICERBU ICERGC ICERPC ICEABS ICEAGC ICEACM ICERBU ICERCU ICERPC ICEABU ICEAOZ ICEAMC ICERBU ICERCU ICERPC ICEABU ICEAOZ ICEAMC ICERBU ICERCU ICERPC ICEABU ICEACC ICEACC ICERCC ICERCC ICEABC ICEACC ICERCC ICERCC ICERCC ICEACC ICEACC ICERCC ICERCC ICERCC ICERCC ICEACC ICEACC ICERCC ICERCC ICERCC ICERCC ICERCC ICEACC ICEACC ICERCC IC	ICEABG	ICEADT	ICFFX2	ICEACX	ICEFDQ	ICFRCT		
ICEABJ ICEAGA ICEREB ICEREJ ICERDT ICEROY ICEABK ICEAGE ICERBC ICEEBK ICERDU ICEROZ ICEABL ICEAGC ICERBE ICEEBL ICEEDX ICERPA ICEABH ICEAGC ICERBE ICEEBL ICEEDX ICERPA ICEABH ICEAGC ICERBE ICEEBL ICERBE ICERPE ICEABN ICEAGE ICEBB ICERBO ICERGO ICERPO ICEABO ICEAGF ICEAGK ICERBO ICERGO ICERPO ICEABO ICEAGF ICEAGK ICERBO ICERGO ICERPO ICEABO ICEAGG ICEAGL ICERBP ICEEGE ICERPE ICEABC ICEAGH ICEAGM ICERBU ICEEGL ICERPO ICEABR ICEAGI ICEAGM ICERBU ICEEGL ICERPO ICEABS ICEAGI ICEAMA ICERBU ICEEGH ICERPO ICEABS ICEAGI ICEAMA ICERBU ICERGU ICERPO ICEABU ICEAOY ICEAMB ICERBU ICERGU ICERPO ICEABU ICEAOY ICEAMC ICEEBX ICEEMA ICEEBN ICEABU ICEAPA ICEAOA ICERBY ICEEMB ICEEBC ICEABU ICEAPA ICEAOA ICERBY ICEEMB ICEEBC ICEABR ICEAPC ICEAOC ICERCT ICEEOA ICEECR ICEABR ICEAPC ICEAOC ICERCT ICEEOA ICEEGG ICEABR ICEAPE ICEAOC ICERCT ICEEOA ICEEGG ICEABR ICEAPF ICEAOC ICERCO ICEEGC ICEEGG ICEABC ICEAPF ICEAOC ICERCO ICEEGO ICEEGON ICEADC ICEAPF ICEAOC ICEECS ICEEGO ICEEGON ICEADC ICEAPF ICEAOC ICEECS ICEEGO ICEEGON ICEADC ICEAPF ICEAOC ICEECS ICEEGO ICEEGON ICEADC ICEAPH ICEAOC ICEECS ICEEGC ICEEGEN ICEADC ICEAPI ICEAOC ICEECS ICEEGO ICEEGON ICEADC ICEAPI ICEAOL ICERCO ICEECG ICEEGEN ICEADC ICEAPI ICEAOL ICERCO ICEECC ICEEGEN ICEADC ICEAPH ICEAOL ICERCO ICEECC ICEEGEN ICEADC ICEACH ICEACH ICEECC ICEECC ICEEGEN ICEADC ICEACH ICEACH ICEECC ICEECC ICEEGEN ICEACH ICEACH ICEACH ICEACH ICEECC ICEACH IC	ICEABH		ICEEX3	ICFFBH	ICERDR	ICEROW		
ICEABK ICEAGE ICEBE ICEBE ICEBU ICEROZ ICEABL ICEAGC ICEBE ICEBU ICEEDX ICERPA ICEABM ICEAGC ICEBE ICEBU ICEEDX ICERPA ICEABM ICEAGC ICEBE ICEBU ICERGE ICERPE ICEABN ICEAGE ICEBE ICEBE ICERGE ICERPC ICEABO ICEAGE ICEAGK ICEBE ICERGO ICERPO ICEABO ICEAGE ICEAGK ICEBE ICERPE ICEABC ICEAGI ICEAGM ICEBE ICERFE ICEABC ICEAGI ICEAGM ICERBU ICERGI ICERPF ICEABS ICEAGI ICEAMA ICEBU ICERGI ICERPG ICEABS ICEAGI ICEAMA ICEBU ICERGI ICERPM ICEABS ICEACI ICEAMA ICEBU ICERGU ICERPM ICEABU ICEAOZ ICEAMC ICEBBX ICERMA ICEBEN ICEABU ICEAOZ ICEAMC ICEBBX ICERMA ICEBEN ICEABU ICEAPA ICEAOA ICERBY ICEFME ICEBEC ICEABN ICEAPE ICEAOA ICERBY ICEFME ICEBCC ICEABN ICEAPC ICEAOC ICERCT ICEFOA ICEBCR ICEABY ICEAPC ICEAOC ICERCT ICEFOA ICEBCB ICEABZ ICEAPE ICEAOC ICERCO ICERCC ICERGC ICEABD ICEAPE ICEAOC ICERCO ICERCO ICEBCB ICEADC ICEAPG ICEAOC ICERCO ICERCO ICEBCA ICEADC ICEAPG ICEAOC ICERCO ICEBCA ICEADC ICEACO ICEACO ICEBCA ICEADC ICEACO ICEACO ICEBCA ICEADC ICEACO ICEACO ICEBCA ICEACO ICEACO ICEBCA ICEACO ICEACO ICEACO ICEBCA ICEACO ICEACO ICEACO ICEBCA ICEACO ICEACO ICEACO ICEBCA ICEACO ICEAC		ICEADX	ICFFBA	ICERBI	ICERDS	ICEROX		
ICEABL ICEAGC ICERBE ICEBUL ICEEDX ICERPA ICEABM ICEAGD ICERBF ICERPS ICERGB ICERPE ICEABM ICEAGE ICERBF ICERBM ICERGC ICERPC ICEABO ICEAGF ICEAGK ICERBO ICERGD ICERPD ICEABO ICEAGF ICEAGK ICERBO ICERGD ICERPD ICEABO ICEAGI ICEAGM ICERBT ICERGF ICERPF ICEABR ICEAGI ICEAGM ICERBU ICERGI ICERPF ICEABR ICEAGI ICEAMA ICERBU ICERGI ICERPF ICEABS ICEAGI ICEAMA ICERBU ICERGU ICERPM ICEABS ICEAGI ICEAMA ICERBU ICERGU ICERPM ICEABU ICEAOZ ICEAMC ICEEBX ICERMA ICERPV ICEABU ICEAOZ ICEAMC ICEEBX ICERMA ICEBBN ICEABU ICEAPA ICEAOA ICERBY ICEFMC ICEBCR ICEABN ICEAPB ICEAOA ICERBY ICEFMC ICEBCR ICEABN ICEAPC ICEAOC ICERCT ICEFOA ICEBCR ICEABN ICEAPC ICEAOC ICERCT ICEFOA ICEBCR ICEABN ICEAPF ICEAOC ICERCV ICERCE ICERGG ICEABR ICEAPF ICEAOC ICERCT ICEFOA ICEBGB ICEABC ICEAPF ICEAOC ICERCF ICERCF ICERGO ICEADD ICEAPF ICEAOC ICERCF ICERCF ICERGO ICEADD ICEAPH ICEAOH ICERCF ICERCF ICERPA ICEADD ICEAPI ICEAOI ICERCF ICERCF ICERPA ICEADG ICEAPI ICEAOI ICERCF ICERCF ICERPA ICEADH ICEAOK ICEADD ICERCF ICERCF ICERPA ICEADH ICEAOK ICEADD ICERCF ICERCF ICERPA		ICFAGA	ICFREB	ICEREJ	ICERDT	-		
ICEABM ICEAGD ICERBF ICERPM ICERGB ICERPE ICEABN ICEAGE ICFRBG ICFREN ICERGC ICERPC ICEABO ICEAGF ICEAGK ICERBC ICERGD ICERPD ICEABO ICEAGF ICEAGK ICERBC ICERPD ICEABP ICEAGG ICEAGL ICERBP ICERGE ICERPE ICEABP ICEAGG ICEAGM ICERBT ICERGF ICERPF ICEABR ICEAGI ICEAGN ICERBU ICERGI ICERPG ICEABS ICEAGI ICEAMA ICERBU ICERGM ICERPM ICEABS ICEAGI ICEAMA ICERBU ICERGM ICERPM ICEABU ICEAOY ICEAMB ICERBU ICERGV ICERPV ICEABU ICEAOZ ICEAMC ICEFBX ICERMA ICEBBN ICEABU ICEAPA ICEAOA ICERBY ICEFME ICEBBC ICEABN ICEAPB ICEAOA ICERBY ICEFMC ICEBCR ICEABN ICEAPC ICEAOC ICERCT ICEFOA ICEBCR ICEABY ICEAPC ICEAOC ICERCT ICEFOA ICEBCR ICEABY ICEAPF ICEAOC ICERCV ICERCE ICEBCB ICEABZ ICEAPF ICEAOC ICERCV ICERCE ICEBCN ICEABC ICEAPF ICEAOC ICERCF ICERCO ICEBCN ICEADC ICEAPF ICEAOC ICERCF ICERCF ICEBCN ICEADC ICEACH ICEACH ICEBCD ICEFCF ICEBCN ICEADC ICEACH ICEACH ICEBCD ICERCF ICEBCN ICEADC ICEACH ICEACH ICEBCD IC	ICEABK	ICEAGE	ICEPBC	ICEFEK	ICERDU	ICFROZ		
ICEABN ICEAGE ICERBG ICEREN ICERGC ICERPC ICEABO ICEAGF ICEAGK ICERBO ICERGD ICERPD ICEABP ICEAGG ICEAGL ICEBBP ICERGE ICERPE ICEABP ICEAGG ICEAGL ICEBBP ICERGE ICERPE ICEABR ICEAGI ICEAGM ICERBU ICERGI ICERPF ICEABR ICEAGI ICEAMA ICERBU ICERGI ICERPG ICEABS ICEAGJ ICEAMA ICERBU ICERGM ICERPM ICEABS ICEACJ ICEAMA ICERBU ICERGV ICERPV ICEABU ICEAOZ ICEAMC ICEEBX ICERMA ICEBBN ICEABU ICEAOZ ICEAMC ICEEBX ICEEMA ICEBBN ICEABU ICEAPA ICEAOA ICERBY ICEEMB ICEBBC ICEABN ICEAPB ICEAOA ICERBZ ICEEMC ICEBCR ICEABN ICEAPC ICEAOC ICERCT ICEEOA ICEBCR ICEABY ICEAPC ICEAOC ICERCT ICEEOA ICEBCB ICEABY ICEAPE ICEAOC ICERCV ICEEGB ICEABZ ICEAPE ICEAOC ICERCV ICEEGB ICEABZ ICEAPE ICEAOC ICERCO ICEEGC ICEADB ICEAPF ICEAOC ICEECT ICEEOA ICEBCN ICEADC ICEAPG ICEAOC ICEECT ICEEOA ICEBCA ICEADC ICEAPG ICEAOC ICEECT ICEBCA ICEADC ICEAPG ICEAOC ICEECT ICEBCA ICEADC ICEAPG ICEAOC ICERCC ICEBCA ICEADC ICEACOC ICEBCA ICEACC ICEACC ICEACC ICEBCA ICEACC I		ICFAGC	ICERBE	ICFFuL	ICEFDX	ICERPA		
TCEABO ICEAGF ICEAGK ICERBO ICERGD ICERPD ICEABP ICEABP ICEABC ICEABL ICERBP ICERGE ICERPE ICEABR ICEAGH ICEAGM ICERBT ICERGF ICERPF ICEABR ICEAGH ICEAGM ICERBU ICERGI ICERPF ICEABR ICEAGI ICEAGM ICERBU ICERGI ICERPM ICEABS ICEAGJ ICEAMA ICERBU ICERGM ICERPM ICEABS ICEACO ICEAMB ICERBU ICERGV ICERPV ICEABU ICEACO ICEAMC ICEEBX ICERMA ICEABN ICEABU ICEABA ICEACO ICERBY ICEEMB ICEABC ICEABR ICEACO ICERCT ICEECA ICEACR ICEABX ICEAPC ICEACO ICERCT ICEECA ICEACR ICEABY ICEACO ICERCT ICEECA ICEACR ICEABS ICEACO ICERCT ICEECA ICEACO IC			ICEFBF	ICERPM	ICERGB			
ICEABP ICEAGG ICEAGL ICERBP ICERGE ICERPE ICEABQ ICEAGH ICEAGM ICERBT ICERGF ICERPF ICEABR ICEAGI ICEAGN ICERBU ICERGL ICERPG ICEABR ICEAGI ICEAMA ICERBU ICERGL ICERPG ICEABS ICEAGJ ICEAMA ICERBU ICERGM ICERPM ICEABT ICEAOY ICEAMB ICERBU ICERGV ICERPV ICEABU ICEAOZ ICEAMC ICEEBX ICERMA ICEBBN ICEABU ICEAPA ICEAOA ICERBY ICEEMB ICEBBC ICEABW ICEAPB ICEAOB ICERBZ ICEEMC ICEBCR ICEABX ICEAPC ICEAOC ICERCT ICEEOA ICEBCJ ICEABX ICEAPC ICEAOC ICERCT ICEEOA ICEBCJ ICEABZ ICEAPE ICEAOC ICERCT ICEECB ICEBCB ICEABZ ICEAPF ICEAOF ICEECF ICEECD ICEBCN ICEADB ICEAPF ICEAOF ICEECF ICEECD ICEBCN ICEADC ICEAPG ICEAOF ICEECB ICEECB ICEADC ICEAPG ICEAOF ICEECB ICEECB ICEADC ICEAPG ICEAOF ICEECB ICEECB ICEADC ICEAPG ICEAOI ICEECB ICEECB ICEADC ICEAPG ICEAOI ICEECB ICEECB ICEADC ICEAPI ICEAOI ICEECB ICEECB ICEECBBN ICEADC ICEAPI ICEAOI ICEECC ICEECB ICEECBBN ICEADC ICEAPI ICEAOI ICEECB ICEECB ICEECBBN ICEADC ICEAPI ICEAOI ICEECC ICEECB ICEECBBN ICEADC ICEAPI ICEAOI ICEECD ICEECB ICEECBBN ICEADC ICEAPI ICEAOI ICEECC ICEECBC ICEECBBN ICEADC ICEACOK ICEBDD ICEECI ICEEDD		_	ICFFBG	ICFREN	ICERGC	ICFRPC		
ICEABQ ICEAGH ICEAGM ICERBT ICERGF ICERPF ICEABR ICEAGI ICEAGN ICERBU ICERGI ICERPG ICEABS ICEAGI ICEAMA ICERBU ICERGI ICERPM ICEABS ICEAGI ICEAMA ICERBU ICERGM ICERPM ICEABT ICEAOY ICEAMB ICERBU ICERGV ICERPV ICEABU ICEAOZ ICEAMC ICEEBX ICERMA ICEBBN ICEABU ICEAPA ICEAOA ICERBY ICEFME ICEBBC ICEABW ICEAPB ICEAGB ICERBZ ICEFMC ICEBCR ICEABX ICEAPC ICEAOC ICERCT ICEGOA ICEBDJ ICEABY ICEAPD ICEAOD ICERCV ICEBCB ICEBGB ICEABZ ICEAPE ICEAOE ICERC6 ICERCC ICEBGC ICEADB ICEAPF ICEAOF ICERC7 ICEBCD ICEBON ICEADC ICEAPG ICEAOG ICERC8 ICERCE ICEBPA ICEADC ICEAPH ICEAOH ICEBC9 ICERCF ICEBPA ICEADG ICEAPI ICEAOI ICERCB ICERCG ICEBPA ICEADG ICEAPI ICEAOI ICERCB ICERCG ICEBPA ICEADG ICEAPI ICEAOI ICERCB ICERCG ICEBPN ICEADG ICEAPJ ICEAOJ ICERCC ICEFCH ICEBBC ICEADH ICEAPK ICEAOK ICEBDD ICERCI ICEBDJ		ICFAGF		ICFRBG	ICERGD	ICERPD		
ICHABR ICEAGI ICEAGN ICEREU ICERGI ICEREGICEABS ICEAGJ ICEAMA ICEREU ICERGM ICEREM ICEABS ICEAGJ ICEAMA ICEREU ICERGM ICEREM ICEABT ICEAOU ICEAMB ICEREW ICERGV ICEREV ICEABU ICEAOZ ICEAMC ICEEBX ICERMA ICEBEN ICEABU ICEAPA ICEAOA ICEREY ICEEME ICEBEC ICEABU ICEAPA ICEAOA ICEREY ICEEME ICEBCR ICEABX ICEAPE ICEAOC ICERCT ICEBOA ICEBCJ ICEABX ICEAPE ICEAOC ICERCT ICEBOA ICEBCJ ICEABY ICEAPE ICEAOC ICERCV ICEBCB ICEBCG ICEABZ ICEAPE ICEAOE ICERC6 ICERCC ICEBCC ICEACC ICEADB ICEAPF ICEAOF ICEBCB ICEBCB ICEBCB ICEACD ICEAPA ICEADD ICEACH ICEBCB ICEBCB ICEBCB ICEAPA ICEADD ICEACH ICEBCB ICEBCB ICEBCB ICEBCB ICEACH ICE			ICEAGL	ICERBP	ICEFGE	ICERPE		
ICEABS ICEAGJ ICEAMA ICERBY ICERGM ICERPM ICEABT ICEAOY ICEAMB ICERBW ICERGY ICERPY ICEABU ICEAOZ ICEAMC ICEEBX ICERMA ICEBBN ICEABU ICEAPA ICEAOA ICERBY ICEEMB ICEBBC ICEABW ICEAPB ICEAGB ICEBBZ ICEEMC ICEBCR ICEABX ICEAPC ICEAOC ICERCT ICEGOA ICEBCJ ICEABY ICEAPD ICEAOD ICERCY ICERCB ICEBCB ICEABZ ICEAPE ICEAOE ICERC6 ICERCC ICEBCC ICEADB ICEAPF ICEAOF ICECC7 ICEBCD ICEBCN ICEADC ICEAPG ICEAOG ICEBCB ICEBCB ICEBCB ICEADC ICEAPG ICEAOG ICEBCB ICEBCB ICEBCB ICEADD ICEAPH ICEAOH ICEBCB ICEBCB ICEBCB ICEADG ICEAPJ ICEAOI ICEBCB ICEBCB ICEBCB ICEADG ICEAPJ ICEAOI ICEBCB ICEBCB ICEBCB ICEADD ICEAPJ ICEAOI ICEBCB ICEBCB ICEBCB ICEADD ICEAPK ICEAOK ICEBDD ICEBCI ICEBDJ	-	ICFAGH	ICEAGM	ICERBT	ICERGE	ICERPF		
ICEABT ICEAOY ICEAMB ICERBW ICERGY ICERPY ICEABU ICEAOZ ICEAMC ICEEBX ICERMA ICEBBN ICEABV ICEAPA ICEAOA ICERBY ICEEMB ICEBBC ICEABW ICEAPB ICEAGB ICEBBZ ICEEMC ICEBCR ICEABX ICEAPC ICEAOC ICERCT ICEGOA ICEBCJ ICEABX ICEAPC ICEAOC ICERCT ICEGOA ICEBCJ ICEABY ICEAPE ICEAOC ICERCY ICERCE ICEGG ICEABZ ICEAPE ICEAOE ICERCE ICERCC ICEGC ICEADB ICEAPF ICEAOF ICEGC ICEGC ICEGGN ICEADC ICEAPG ICEAOG ICEGCB ICEGCB ICEBCB ICEADC ICEAPG ICEAOG ICEGCB ICEGCB ICEGBA ICEADC ICEAPH ICEAOH ICEGCB ICEGCB ICEGBN ICEADC ICEAPI ICEAOI ICEBCB ICEGCB ICEGBN ICEADG ICEAPJ ICEAOI ICEBCB ICEGCB ICEGBN ICEADG ICEAPJ ICEAOK ICEBCD ICEGCH ICEGBC ICEADH ICEAOK ICEBCD ICEGCI ICEGDJ		ICEAGI	ICEAGN	ICERBU	ICERGL	ICFRPG		
ICEABU ICEAOZ ICEAMC ICEEBX ICERMA ICEBEN ICEABV ICEAPA ICEAOA ICEEBY ICEEME ICEBEC ICEABV ICEAPB ICEAGB ICEBBZ ICEEME ICEBEC ICEABW ICEAPB ICEAOC ICERCT ICEEOA ICEBCJ ICEABY ICEAPD ICEAOC ICERCT ICEEOA ICEBCJ ICEABY ICEAPE ICEAOE ICERCE ICERCE ICEBCG ICEABZ ICEAPE ICEAOE ICERCE ICERCE ICEBCJ ICEADB ICEAPF ICEAOF ICEECT ICEBCJ ICEBCJ ICEADC ICEAPG ICEAOG ICEBCB ICERCE ICEBPA ICEADD ICEAPH ICEAOH ICEBCJ ICERCF ICEBPA ICEADC ICEAPI ICEAOI ICEBCJ ICEBCG ICEBBN ICEADG ICEAPJ ICEAOJ ICEBCJ ICEBCJ ICEBCJ ICEADH ICEAPK ICEAOK ICEBDD ICEBCI ICEBDJ			ICEAMA	ICEREV	ICERGM	ICERPM		
ICEABV ICEAPA ICEAOA ICERBY ICEEME ICEBBC ICEABW ICEAPB ICEAGB ICEBBZ ICEEMC ICEBCR ICEABX ICEAPC ICEAOC ICERCT ICEEOA ICEBDJ ICEABY ICEAPD ICEAOD ICERCV ICEBCB ICEBGB ICEABZ ICEAPE ICEAOE ICERC6 ICERCC ICEBGC ICEADB ICEAPF ICEAOF ICEEC7 ICEBCD ICEBGN ICEADC ICEAPG ICEAOG ICEBCB ICEBCB ICEBCB ICEADD ICEAPH ICEAOH ICEBCB ICEBCB ICEBCB ICEADD ICEAPH ICEAOH ICEBCB ICEBCB ICEBCB ICEADG ICEAPJ ICEAOI ICEBCB ICEBCB ICEBCB ICEADG ICEAPJ ICEAOI ICEBCB ICEBCB ICEBCB ICEADH ICEAPK ICEAOK ICEBDD ICEBCI ICEBDJ		ICEYOA	ICEAMB	ICERBW	ICERGV	ICERPV		
ICEABW ICEAPB ICEAGB ICEBBZ ICEFMC ICEACR ICEABX ICEAPC ICEAOC ICERCT ICEBOA ICEBDJ ICEABY ICEAPD ICEAOD ICERCV ICEBCB ICEBGB ICEABZ ICEAPE ICEAOE ICERC6 ICERCC ICEBGC ICEADB ICEAPF ICEAOF ICEBC7 ICEBCD ICEBGN ICEADC ICEAPG ICEAOG ICEBC8 ICEBCB ICEBCA ICEADD ICEAPH ICEAOH ICEBC9 ICEBCF ICEBPA ICEADD ICEAPH ICEAOH ICEBC9 ICEBCF ICEBPA ICEADG ICEAPJ ICEAOI ICEBCB ICEBCB ICEBCB ICEADG ICEAPJ ICEAOK ICEBCD ICEBCH ICEBBC ICEADH ICEAPK ICEAOK ICEBDD ICEBCI ICEBDJ			_	ICEFBX	ICERMA			
ICEABX ICEAPC ICEAOC ICERCT ICEBOA ICEBDJ ICEABY ICEAPD ICEAOD ICERCY ICEBOB ICEBGB ICEABZ ICEAPE ICEAOE ICERC6 ICERCC ICEBGC ICEADB ICEAPF ICEAOF ICEBC7 ICEBCD ICEBGN ICEADC ICEAPG ICEAOG ICEBC8 ICEBCB ICEBCB ICEADD ICEAPH ICEAOH ICEBC9 ICEBCF ICEBPA ICEADD ICEAPH ICEAOH ICEBC9 ICEBCF ICEBPM ICEADG ICEAPJ ICEAOI ICEBCB ICEBCB ICEBCB ICEADG ICEAPJ ICEAOK ICEBCD ICEBCH ICEBBC ICEADH ICEAPK ICEAOK ICEBDD ICEBCI ICEBDJ			ICEAGA	ICERBY	ICEFMB			
ICEABY ICEAPE ICEAOD ICERCY ICEROE ICEAGE ICEABZ ICEAPE ICEAOE ICERC6 ICERCC ICEAGC ICEADB ICEAPF ICEAOF ICEC7 ICERCD ICEAGN ICEADC ICEAPG ICEAOG ICERC8 ICERCE ICEAPA ICEADD ICEAPH ICEAOH ICERC9 ICERCF ICEAPA ICEADE ICEAPI ICEAOI ICERC6 ICERC6 ICEAPH ICEADG ICEAPJ ICEAOI ICERC6 ICEACH ICEADG ICEAPJ ICEACJ ICERC6 ICEFCH ICEAPBC ICEADH ICEAPK ICEAOK ICEBDD ICERCI ICEADJ				ICERBZ	ICFFMC			
ICEABZ ICEAPE ICEAOE ICERC6 ICERCC ICERGC ICEADB ICEAPF ICEAOF ICERC7 ICERCD ICE8ON ICEADC ICEAPG ICEAOG ICERC8 ICERCE ICE8PA ICEADD ICEAPH ICEAOH ICERC9 ICERCF ICE8PM ICEADE ICEAPI ICEAOI ICERCB ICERCG ICE9BN ICEADG ICEAPJ ICEACJ ICERCC ICEFCH ICE9BC ICEADH ICEAPK ICEAOK ICEBDD ICERCI ICE9DJ				ICFRCT	ICERUA	ICF8DJ		
ICEADB ICEAPF ICEAOF ICEFC7 ICEFCD ICE8ON ICEADC ICEAPG ICEAOG ICEFC8 ICEFCE ICE8PA ICEADD ICEAPH ICEAOH ICEFC9 ICEFOF ICE8PM ICEADE ICEAPI ICEAOI ICEFC6 ICEFCF ICE9BN ICEADG ICEAPJ ICEACJ ICEFC ICEFCH ICE9BC ICEADH ICEAPK ICEAOK ICERDD ICEFCI ICE9DJ					-			
ICEADC ICEAPG ICEAOG ICERCB ICERCE ICEAPA ICEADD ICEAPH ICEAOH ICERC9 ICEROF ICEAPM ICEADE ICEAPI ICEAOI ICERCB ICERCG ICEAPN ICEADG ICEAPJ ICEACJ ICERCC ICEFCH ICEAPBC ICEADH ICEAPK ICEAOK ICERDD ICERCI ICEADJ								
ICEADD ICEAPH ICEAOH ICERC9 ICEROF ICE8PM ICEADE ICEAPI ICEAOI ICERDE ICERCG ICE9BN ICEADG ICEAPJ ICEACJ ICERCC ICEFCH ICE9BC ICEADH ICEAPK ICEAOK ICEBDD ICERCI ICE9DJ						ICE80N		
ICEADE ICEAPI ICEAOI ICERDE ICERCG ICE9BN ICEADG ICEAPJ ICEACJ ICERCC ICEFCH ICE9BC ICEADH ICEAPK ICEAOK ICEBDD ICERCI ICE9DJ				ICERC8				
ICEADG ICEAPJ ICEACJ ICEEDC ICEECH ICE9BC ICEADH ICEAPK ICEADK ICEEDD ICEECI ICE9DJ								
ICEADH ICEAPK ICEAOK ICEADD ICERCI ICEADJ								
ICEADI ICEAPL ICEAON ICERDE ICEFOJ ICEGGE								
ICEADJ ICEAP1 ICEAOP ICERDG ICEROK ICEGGC						ICE9GE		

Figure 11. Conventional Technique Modules

Load Module	Object Mod	lules	
ICERCM	ICERCM ICERCU ICERCQ ICERCD ICERC3	ICERCH ICERCX ICERCP ICERCI ICERC2	ICERCN ICEBGB ICERCS ICERCL
ICERCZ	ICERCZ ICERCU ICERC1 ICEAOL ICEAOM	ICEAO4 ICEAO5 ICERC4 ICERCK ICE8CK	ICERCJ ICEAO1 ICEAO2

Figure 12. Object Modules in the Two Overlay Load Modules

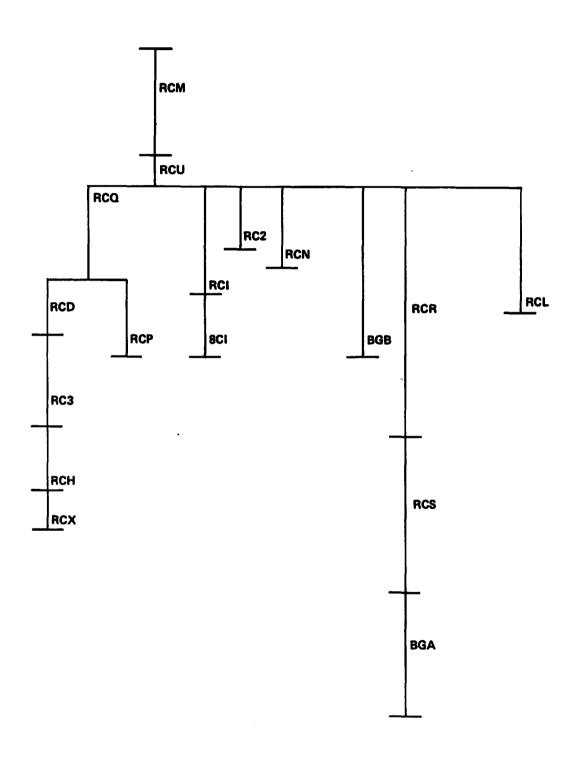


Figure 13. (1 of 6) Load Modules With Conventional Techniques
Phase 0, Initialization, Stage 2 (Definition)
Overlay Structure

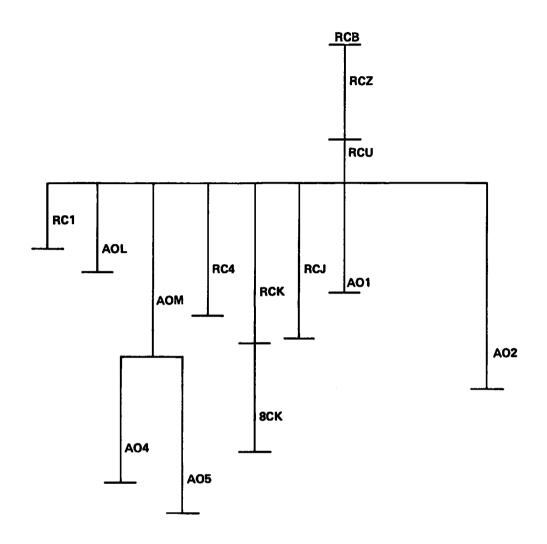


Figure 13. (2 of 6) Load Modules With Conventional Techniques Phase 0, Initialization, Stage 2 (Optimization) Overlay Structure

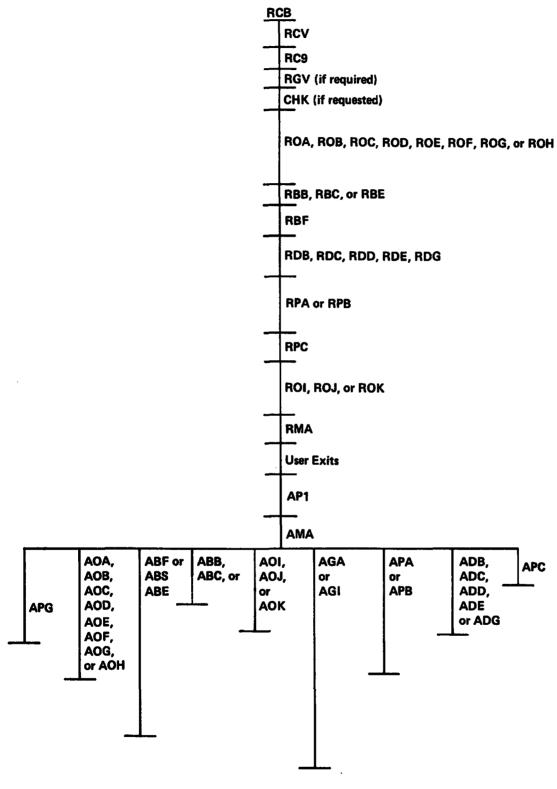


Figure 13. (3 of 6) Load Modules with Conventional Techniques
Phase 1, BALN and POLY Techniques.
Overlay Structure.

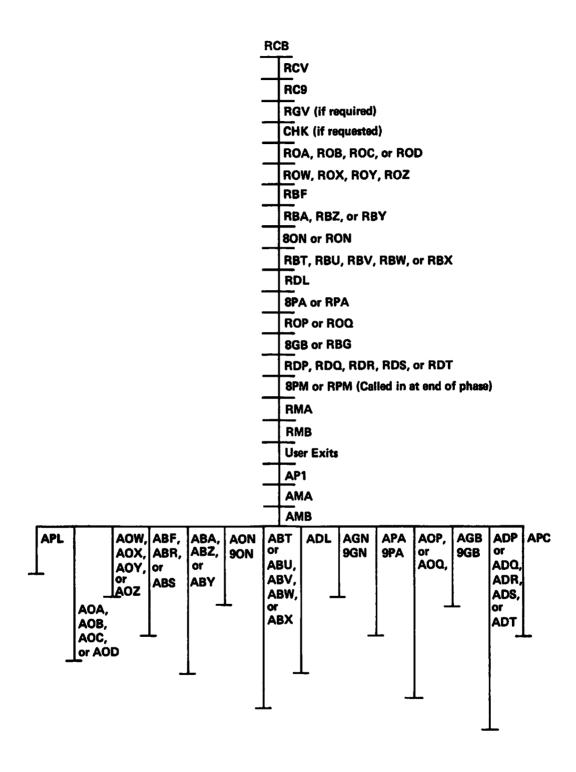


Figure 13. (4 of 6) Load Modules with Conventional Techniques Phases 1 and 2, OSCL and CRCX Techniques. Dynamic.

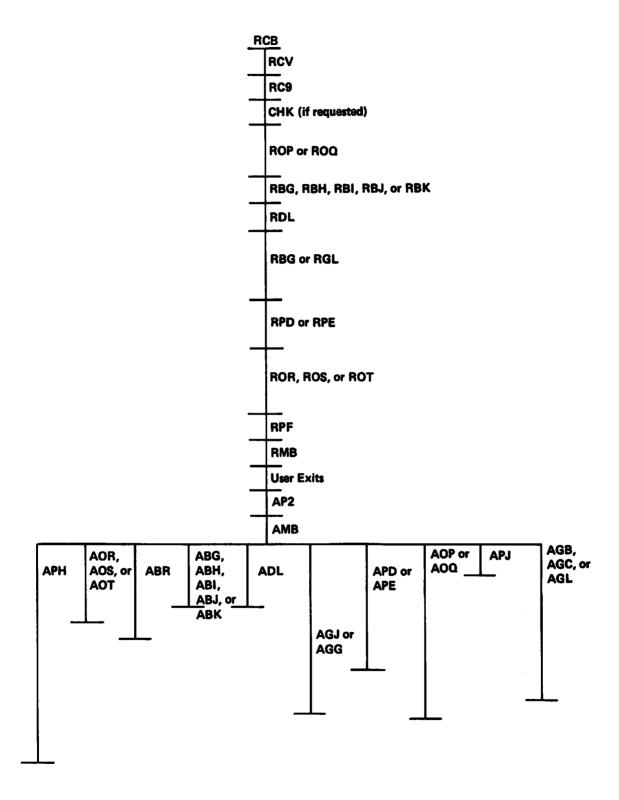


Figure 13. (5 of 6) Load Modules with Conventional Techniques
Phase 2, BALN and POLY Techniques.
Dynamic.

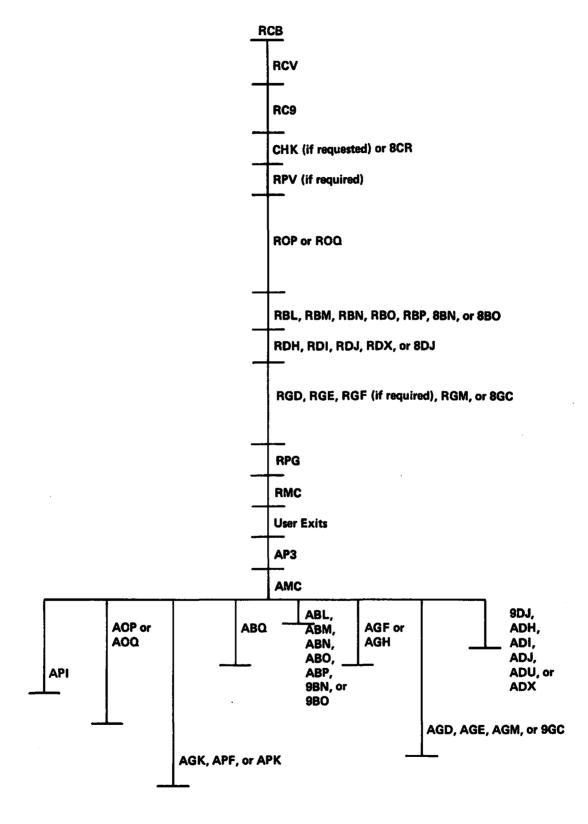


Figure 13. (6 of 6) Load Modules with Conventional Techniques Phase 3. Dynamic.

Storage Layouts

| BLOCKSET, PEERAGE, AND VALE TECHNIQUES

| Storage is allocated as follows:

- Phase 0 issues a GETMAIN primarily for the communications area, COMMON or COMMA, and the extract/restore code area, which is variable in length.
- 2. If SORTWK data sets have been defined, a GETMAIN is issued for the necessary I/O area. Storage is also allocated to the Checkpoint/ Restart track pool if necessary (though not for Blockset).
- 3. End of Phase 0 frees whatever part of the first area is not needed by COMMON or COMMA and the extract/restore code area.
 - 4. Phases 1 and 3 begin by getting space for buffers for SORTIN and SORTOUT.
 - 5. Each of the Phases 1-3 gets the remaining required storage by issuing a series of variable GETMAINS until it reaches SIZE or MAXLIM. At the end of the phase it frees this storage again.

OTHER TECHNIQUES

The storage allocation technique used is such that main storage is first allocated for necessary or required areas (proceeding from the largest to the smallest), and then for the optional areas (in priority order). This is accomplished by issuing variable-type requests for main storage. The minimum request is equal to the size of the largest necessary area; and the maximum, equal to the total size of all areas.

If additional main storage is needed, the unused portion of the previous allocation is made available for subsequent allocations. The minimum request becomes equal to the size of the area next-in-line to be allocated; and the maximum, equal to the total size of all unallocated areas. (This process is continued until either all areas are allocated, or the minimum request cannot be satisfied.)

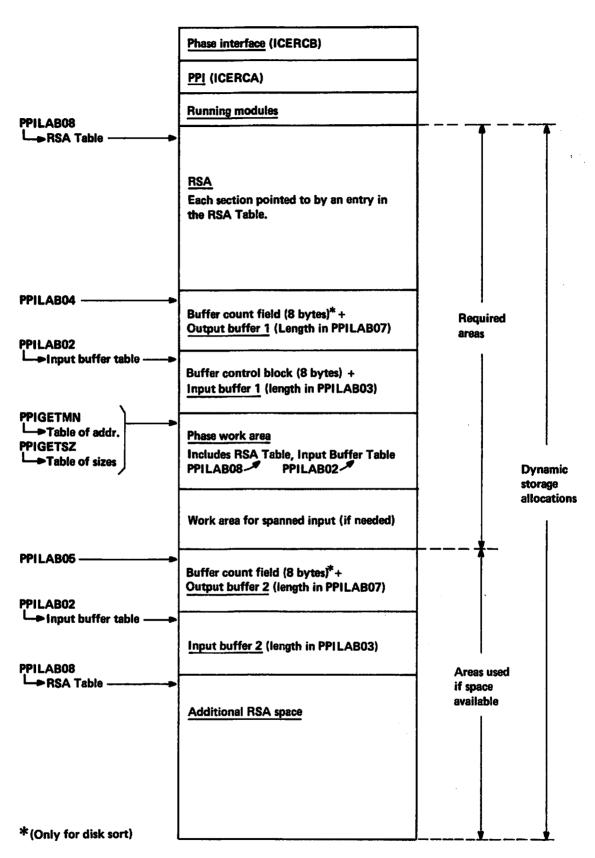


Figure 14. (1 of 4) Storage Layouts for Conventional Techniques Phase 1, BALN and POLY Techniques

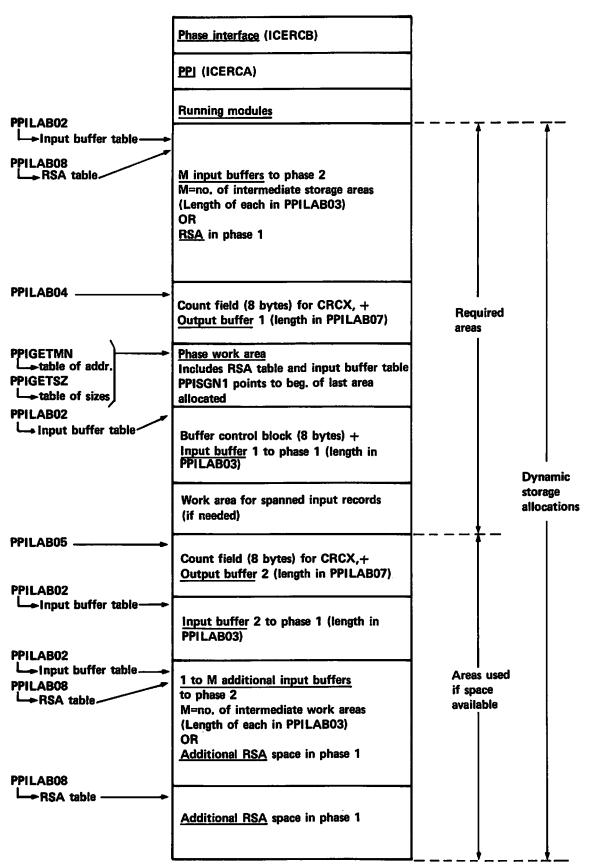


Figure 14. (2 of 4) Storage Layouts for Conventional Techniques
Phases 1 and 2, CRCX and OSCL Techniques

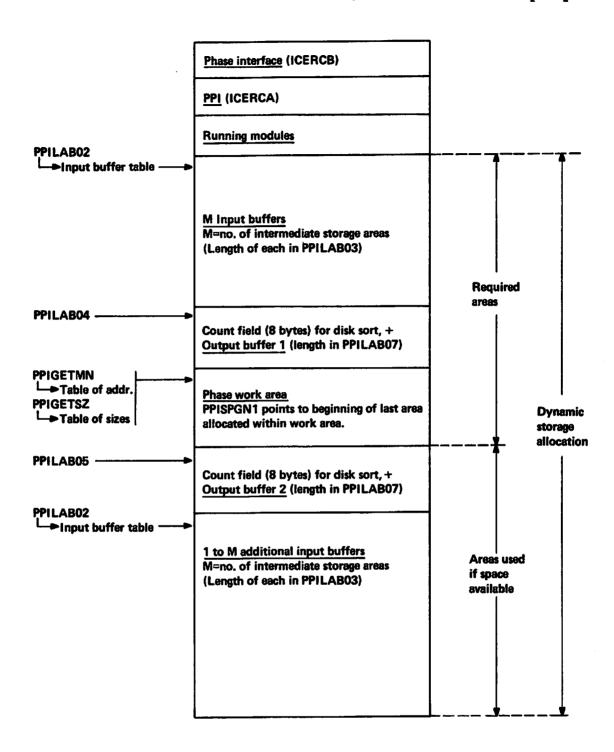


Figure 14. (3 of 4) Storage Layouts for Conventional Techniques Phase 2, POLY and BALN Techniques

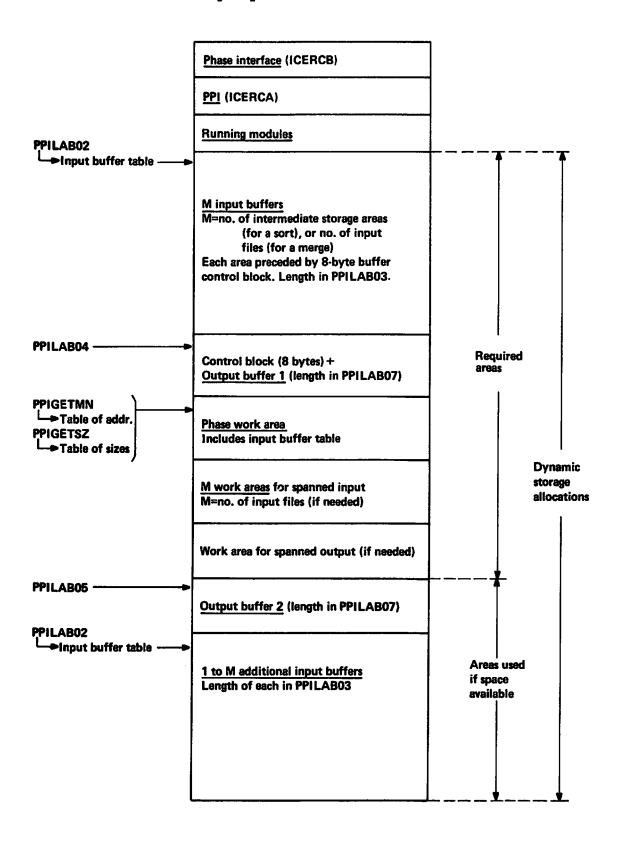


Figure 14. (4 of 4) Storage Layouts for Conventional Techniques
Phase 3, All Techniques

Section 4. Directories

This section lists all module names: first Blockset, then Peerage and Vale, then conventional technique modules. Each list is in alphabetical order. Cross references are given to other parts of the publication.

Blockset Directory

This directory lists Blockset object modules. Their names are in the form ICEmod, where mod is the three or four character identification of the module. Only these last characters are listed. The list also gives a reference to the Method of Operation diagrams (MO diag) provided in Section 2.

Module Name	Used in Phase:	FLR or VLR	MO Diag	Purpose
AM1 CKUT COBU COBV CYLN	0 3 3 1 1,2,3	FLR/VLR FLR FLR VLR FLR/VLR	B3.3 B3.3 B3.1	Supply installed defaults Sortout check Exit E35 interface for invoked sort Exit E15 with no SORTIN interface Cylinder selection
DEVT DYNA	A11 0	FLR/VLR FLR/VLR	D3.1	Device table Dynamic allocation of SORTWK (MVS only)
EXII EXIK EXIN EXIO EXIT E15B E35B	1 2 3 3 1 3	FLR FLR/VLR FLR FLR FLR FLR	B3.1 B3.2 D3.1 B3.3 B3.3 B3.1 B3.3	Phase 1 exit Phase 2 exit Phase 0 exit SMF write and final exit from sort Final exit point Exit E15 interface Exit E35 interface
FAUL FILI FILK FILO FORM FVLI FVLN FVLN	012301203	FLR FLR FLR FLR VLR VLR VLR VLR VLR	D3.1 B3.2 B3.3 D3.1 B3.1 B3.2 D3.1 B3.3	Initialization values Load module filler Load module filler Load module filler Control field format tables Input phase load module filler Key Phase load module filler Phase 0 load module filler Output phase load module filler
GENV	1	VLR	B3.1	Point input code generation
INIB INIC INID INIO IPUB IPUT IPVB IPVM IPVM	0 0 0 1 1 1 1	FLR/VLR FLR/VLR FLR/VLR FLR FLR FLR VLR VLR VLR	D3.1 D3.1	Control statement handling Generate extract/restore code Initialize move and tree code Transfer conditions for other techn. Phase 1 I/O handling Handle records and indexes Initialize for Phase 1 Input phase I/O handler Handles input records and indexes Initialize for input phase

FLR = fixed-length record VLR = variable-length record

Licensed Material--Property of IBM
Page of LY33-8042-6, As Updated 31 March 1981, By TNL LN20-9345

Module Name	Used in Phase:	FLR or VLR	MO Diag	Purpose
KPUA KPUB KPUV KPUV KPVA KPVB KPVV KPVV	NNNNNNNNN	FLR FLR FLR FLR VLR VLR VLR VLR	B3.2 B3.2 B3.2 B3.2 B3.2 B3.2 B3.2 B3.2	Virtual block construction Phase 2 I/O handling Virtual block construction Initialize for Phase 2 or 3 Virtual block construction Virtual block construction Key phase I/O handler Key phase writeback merge Initialize for key and output phases Virtual block construction
LIMI	3	FLR	B3.2	Writeback limit
MAN MESI MESS MESV	0 1 All	FLR/VLR FLR FLR/VLR VLR	D3.1	Handle parameter list and DD statements Handle Phase 1 messages Message handling Handle input phase messages
MSGI MSGK MSGN MSGO	1 2 0 3	FLR/VLR FLR/VLR FLR/VLR FLR/VLR		Phasa 1 messages Phase 2 messages Phase 0 messages Phase 3 messages
OPUA OPUB OPVA OPVB OPVT OXIV	55555555	FLR FLR VLR VLR VLR VLR VLR	B3.3 B3.3 B3.3 B3.3 B3.3 B3.3	Phase 3 subroutines Phase 3 I/O handling Handle records and indexes Output phase subroutines Output phase I/O handler Main output driver Output exit and message entry points SMF write and final exit from sort
SUBS	1,3 1,3	FLR VLR		Suballocate main storage Suballocate main storage
XPUT	1,2,3	FLR		Load user exits
15V 35VL	1 3	VLR VLR	B3.1 B3.3	Exit E15 with SORTIN interface Exit E35 interface
FLR = fi	xed-length ariable-len	n record	<u></u>	

110

Peerage and Vale Directory

This directory lists Peerage and Vale load modules. Their names are in the form ICEmod, where mod is the three or four character identification of the module. Only these last characters are listed. The list also gives a reference to the Method of Operation diagrams (MO diag) provided in Section 2.

	Module Name	Used in Phase:	MO Diag	Purpo se
	AM1 BUG	0 All		Supply installed defaults Handle formatted dumps
	CKP	2 All		Prepare for checkpoint and/or release unused work space Source code for ICECOMMA
i	CRE	1	P3.1	(formatted dump) Manage Phase 1
	CRO DEC	1	D3.2	Final output in Phase 1
1	DED DEF	0	D3.2 D3.2	Manage Phase 0 (part 1)
	DEG	0	D3.2, D3.3, P3.1	Manage Phase 0 (part 2)
l i	DEV	0	D3.3	Manage Phase 0 (part 3)
1	ERR FIX (SVS/VS1)	0,1,3 0,1		Handle VSAM I/O errors Page fixing and freeing (SVC)
	FIXO (OS) LIM LIV	0,1 3 3	P3.4 P5.31	DEB modification for I/O chaining (SVC) Manage Phase 3 Handle VSAM output in Phase 3 (Peerage only)
1	Man Mon OBS	0 All 0	D3.1	Handle transfer from Blockset Overall program management Open for SORTIN/SORTOUT EXCP
	PAR RED	2 2	P3.2 P3.3	Manage Phase 2 (partitioning) Manage Phase 2 (merging)
	VED	2	P3.5	Manage Phase 2 (merging, with variable length or very long records)
	VEE VIM	2 3	P3.5 P3.6	Take E25 if specified Manage Phase 3 (variable length or very long records)
į	VIP	3	Ì	Initialize Phase 3
į	VRE	1	P3.1 {	Manage Phase 1 (variable length and very long records)
į	VRN	1	P3.1 {	Manage Phase 1 (variable length or very long records)
į	VRO	1		Final output in Phase 1
	XCP	0		Build channel programs for SORTIN and SORTOUT

Directory for Conventional Techniques

The names are listed alphabetically. All CSECTS are included, plus any other labels referenced in other sections of this publication.

EXPLANATION OF COLUMN HEADINGS

CSECT Name

Naming conventions are described in Appendix D.

Each CSECT name consists of six characters, the first three of which are always ICE. For easy reference, the names are listed by their last three characters only. In Phases 1, 2 and 3 those which begin with 'A' or '9' are generally assignment modules, and those which begin with 'R' or '8' are running modules. With a few exceptions, module Axy will carry out assignment for module Rxy, and so on.

Used in Phase:

The phases are:

- 0 (Def) Definition
 0 (Opt) Optimization
- Sort (only for sorting applications)
- Intermediate merge (only for sorting applications)
- 3 Final merge

The first and last phases have some alternative modules for sorting applications and for merge-only applications, which are indicated as follows:

- 3 Sort Used in Phase 3 of a sorting application
- 3 Merge Used in the merge phase of a merge-only application
- 3 All Can be used in this phase for all applications

The words 'sort', 'merge' and 'all' are used with 0 (Def) and 0 (Opt) in the same way.

MO Diagram

Provides a cross reference to the relevant diagram in the Method of Operation section (Section 2).

Purpose

The major purpose of the CSECT is briefly described. Keep in mind the notes on modules beginning with 'A' and '9' (above, under 'CSECT Name').

- No Mult Means it is not;
- Extract Means an extract routine is used to compare control fields.

	Used in Phase:	MO Diag	Purpose
	4		andr a again black was weening
ABA	1		CRCX & OSCL: block var. records
ABB	1		Block fixed-length records with in-line move
ABC	1		Block fixed-length records with link to multiple move
ABE	1		Block var. records with move
ABF	1		Move var. records
ABG	2		Block or deblock fixed-length records with in-line move (no exits)
ABH	2		Block or deblock fixed-length records with link to multiple move (no exits)
ABI	2		Block or deblock var. records (no exits)
ABJ	2		Block or deblock fixed-length records (exits used)
ABK	2		Block or deblock var. records (exits used)
ABL	3 all		Block fixed-length records with in-line move (no exits)
ABM	3 all		Block fixed-length records (exits used)
ABN	3 all		Block var. records (no exits)
ABO	3 all		Block var. records (exits used)
ABP	3 all		Block fixed-length records with link to multiple move
ABQ	3 all		Move generator for fixed-length records
ABR	2	N3.2	Move generator for fixed-length records
ABS	1	N4.11	Move generator for fixed-length records
ABT	2		CRCX & OSCL: block/deblock fixed-length records up to 256 bytes (no exits)
ABU	2		CRCX & OSCL: block/deblock fixed-length records >256 bytes (no exits)
ABV	2		CRCX & OSCL: block/deblock var. records (no exits)
ABW	2		CRCX & OSCL: block/deblock fixed-length records (exits used)
ABX	2		CRCX & OSCL: block/deblock var. records (exits used)
ABY	1		CRCX & OSCL: block fixed-length records up to 256 bytes
ABZ	1		CRCX & OSCL: block fixed-length records >256 bytes
ADB	1		Deblock fixed-length records with in-line move (no exits)
ADC	1		Deblock fixed-length records with link to multiple move (no exits)
ADD	1		Deblock fixed-length records (exits used)
ADE	1	•	Deblock var. records (exits used)
ADG	1		Deblock var. records (no exits)
ADH	3 sort		Deblock fixed-length records
ADI	3 sort		Deblock var. records
ADJ ADL	3 merge 2		Deblock (no exits) Build tables in input buffer addresses
ADP	1		CRCX & OSCL: deblock fixed-length records up to 256 bytes (no exits)

>256 bytes (no exits) CRCX & OSCL: deblock fixed-length records (exits used) ADS 1 CRCX & OSCL: deblock var. records (exits used) ADT 1 CRCX & OSCL: deblock var. records (no exits) ADT 1 CRCX & OSCL: deblock var. records (no exits) ADU 3 merge Deblock (exits used) ADX 3 sort Deblock for read-forward tape, fixed-length records AGA 1 N3.1 For tape, generate DCBs, IOBs, DCB addresses AGB 2 Read tape backward AGC 2 Read disk AGG 3 sort Read disk AGF 3 merge Open files AGG 2 N3.2 For tape, generate DCBs, IOBs, DCB addresses AGG 3 sort Read disk AGG 3 sort Read disk AGF 3 merge Open files (and call checkpoint module)	CSECT Name	Used in Phase:	MO Diag	Purpose
ADR 1 CRCK & OSCL: deblock fixed-length records (exits used) ADS 1 CRCX & OSCL: deblock var. records (exits used) ADT 1 CRCX & OSCL: deblock var. records (exits used) ADT 1 CRCX & OSCL: deblock var. records (no exits) ADD 3 merge Deblock (exits used) ADD 3 merge Deblock (exits used) AGA 1 N3.1 For tape, generate DCBs, IOBs, DCB addresses Read tape backward AGC 2 Read disk AGG 3 sort Read disk AGG 3 merge Open files AGG 1 N3.1 For tape, generate DCBs, IOBs, DCB addresses AGG 2 N3.2 For tape, generate DCBs, IOBs, DCB addresses AGG 1 N3.1 For disk, generate DCBs, IOBs, DCB addresses AGG 2 N3.2 For disk, generate DCBs, IOBs, DCB addresses AGG 3 sort N3.3 For disk, generate DCBs, IOBs, DCB addresses AGG 3 sort N3.3 For disk, generate DCBs, IOBs, DCB addresses AGG 4 N3.1 OSCL: generate DCBs and IOBs Contain messages for phase 1 assignment modules AMA 1 N3.1 OSCL: generate DCBs and IOBs Contain messages for phase 2 assignment modules AMA 1 Sort insed-length records, 'multiple' (not POLY) Sort fixed-length records, 'multiple' (not POLY) Sort var. records, 'multiple' (not POLY) AOC 1 Sort algorithm AOA 1 POLY: sort fixed-length records, 'multiple' POLY: sort var. records, 'single' POLY: sort var. records, 'single' POLY: sort var. records, 'multiple' AOC 1 BALM (tape) sort algorithm POLY sort algorithm AOC 2 6 3 all Merge, 'multiple' Merge, 'multiple' Merge, 'single'	ADQ	1		
ADT 1 CRCX & OSCL: deblock var. records (no exits) ADU 3 merge ADX 3 sort Deblock (exits used) Deblock for read-forward tape, fixed-length records AGA 1 N3.1 For tape, generate DCBs, IOBs, DCB addresses AGB 2 Read dape backward AGC 2 Read disk AGG 3 sort Read disk AGG 3 merge Open files AGG 2 N3.2 For tape, generate DCBs, IOBs, DCB addresses AGG 3 merge Open files (and call checkpoint module) AGG 1 N3.1 For disk, generate DCBs, IOBs, DCB addresses AGG 2 N3.2 For disk, generate DCBs, IOBs, DCB addresses AGG 3 sort N3.3 For disk, generate DCBs, IOBs, DCB addresses AGG 2 N3.2 For disk, generate DCBs, IOBs, DCB addresses AGG 2 N3.2 For disk, generate DCBs, IOBs, DCB addresses AGG 3 sort For tape, read forward AGN 3 sort For tape, read forward AGN 1 N3.1 OSCL: generate DCBs and IOBs Contain messages for phase 1 assignment modules Contain messages for phase 2 assignment modules Contain messages for phase 3 assignment modules Contain messages for phase 4 assignment modules Contain messages for phase 5 assignment modules Contain messages for phase 6 assignment modules Contain messages for phase 6 assignment modules Contain messages for phase 7 assignment modules Contain messages for phase 8 assignment modules Contain messages for phase 1 assignment modules Contain	ADR	1		CRCX & OSCL: deblock fixed-length records
ADT 3 merge Deblock (exits used) ADX 3 sort Deblock (exits used) ADX 3 sort Deblock (exits used) ADX 3 sort Deblock for read-forward tape, fixed-length records AGA 1 N3.1 For tape, generate DCBs, IOBs, DCB addresses Read dape backward AGE 2 Read dape backward AGE 3 sort Read disk AGG 3 merge Open files AGG 2 N3.2 For tape, generate DCBs, IOBs, DCB addresses AGG 3 sort Open files AGG 1 N3.1 For disk, generate DCBs, IOBs, DCB addresses AGG 1 N3.1 For disk, generate DCBs, IOBs, DCB addresses AGG 2 N3.2 For disk, generate DCBs, IOBs, DCB addresses AGG 2 N3.2 For disk, generate DCBs, IOBs, DCB addresses AGG 2 N3.3 For disk, generate DCBs, IOBs, DCB addresses AGG 2 N3.2 For disk, generate DCBs, IOBs, DCB addresses AGG 2 N3.3 For disk, generate DCBs, IOBs, DCB addresses AGG 2 N3.3 For disk, generate DCBs, IOBs, DCB addresses AGG 2 N3.2 For tape, read forward AGN 1 N3.1 OSCL: generate DCBs and IOBs Contain messages for phase 1 assignment modules AMB 2 Contain messages for phase 2 assignment modules AMC 3 all N3.1 OSCL: generate DCBs and IOBs Contain messages for phase 3 assignment modules AMC 3 all Cost generate DCBs and IOBs Contain messages for phase 3 assignment modules AMC 3 all Supply installed defaults Sort fixed-length records, 'multiple' (not POLY) AOC 1 Sort var. records, 'multiple' (not POLY) AOC 1 Sort var. records, 'single' (not POLY) AOC 1 Sort var. records, 'single' AOC 1 POLY: sort fixed-length records, 'multiple' AOC 1 POLY: sort fixed-length records, 'multiple' AOC 1 POLY: sort var. records, 'single' AOC 2 POLY: sort algorithm AOC 0 (Opt) all D3.4 Generate extract routine AOC 2 BALN (tape) merge algorithm AOC 2 POLY merge algorithm AOC 2 POLY merge algorithm	ADS	1		
ADX 3 sort Deblock for read-forward tape, fixed-length records AGA 1 N3.1 For tape, generate DCBs, IOBs, DCB addresses Read tape backward AGE 2 Read disk AGG 3 sort Read disk AGF 3 merge Open files AGG 2 N3.2 For tape, generate DCBs, IOBs, DCB addresses AGH 3 sort Open files AGG 2 N3.2 For disk, generate DCBs, IOBs, DCB addresses AGH 3 sort N3.1 For disk, generate DCBs, IOBs, DCB addresses AGG 2 N3.2 For disk, generate DCBs, IOBs, DCB addresses AGG 2 N3.2 For disk, generate DCBs, IOBs, DCB addresses AGG 2 N3.3 For disk, generate DCBs, IOBs, DCB addresses AGG 2 N3.2 For disk, generate DCBs, IOBs, DCB addresses AGG 2 N3.3 For disk, generate DCBs, IOBs, DCB addresses AGG 2 N3.3 For disk, generate DCBs, IOBs, DCB addresses AGG 2 N3.3 For disk, generate DCBs, IOBs, DCB addresses AGG 2 N3.2 For tape, read forward AGN 1 N3.1 OSCL: generate DCBs and IOBs Contain messages for phase 1 assignment modules Contain messages for phase 2 assignment modules AMB 2 Contain messages for phase 3 assignment modules AMC 3 all Contain messages for phase 3 assignment modules Contain messages for phase 3 assignment modules AMC 3 all Supply installed defaults Sort fixed-length records, 'multiple' (not POLY) AOC 1 Sort var. records, 'multiple' (not POLY) Sort var. records, 'single' AOG 1 POLY: sort fixed-length records, 'multiple' POLY: sort var. records, 'multiple' AOG 1 POLY: sort var. records, 'single' AOG 2 POLY: sort algorithm AOG 1 POLY: sort algorithm AOG 2 POLY: single' AOG 2 POLY: single' AOG 2 POLY: single' AOG 2 POL	ADT	1		
AGA 1 N3.1 For tape, generate DCBs, IOBs, DCB addresses Read tape backward Read disk Read tape backward Read disk Read tape backward Read disk Read disk Read tape backward Read disk Read	ADU	3 merge		
AGE 2 AGC 2 AGC 3 sort Read disk AGF 3 sort Read Green Bore Read Read Read Read Read Read Read Rea	ADX	3 sort		
AGC 2 AGB 3 sort Read disk AGF 3 merge Open files AGG 2 AGG 2 N3.2 For tape, generate DCBs, IOBs, DCB addresses Open files (and call checkpoint module) AGI 1 N3.1 For disk, generate DCBs, IOBs, DCB addresses AGI 2 N3.2 For disk, generate DCBs, IOBs, DCB addresses AGI 2 N3.3 For disk, generate DCBs, IOBs, DCB addresses AGI 2 AGM 3 sort AGM 1 N3.1 OSCL: generate DCBs, IOBs, DCB addresses AGI 2 AGM 1 N3.1 OSCL: generate DCBs IOBs, DCB addresses AGI 2 AGM 1 AGM 1 AGM 1 AGM 1 AGM 1 AGM 2 Contain messages for phase 1 assignment modules Contain messages for phase 2 assignment modules Contain messages for phase 3 assignment modules Contain messages for phase 3 assignment modules AGM 3 AGM 1 AG	AGA		N3.1	For tape, generate DCBs, IOBs, DCB addresses
AGD 3 sort Read tape backward AGF 3 sort Read disk AGF 3 merge Open files AGG 2 N3.2 For tape, generate DCBs, IOBs, DCB addresses Open files (and call checkpoint module) AGI 1 N3.1 For disk, generate DCBs, IOBs, DCB addresses AGK 2 N3.2 For disk, generate DCBs, IOBs, DCB addresses AGK 3 sort N3.3 For disk, generate DCBs, IOBs, DCB addresses AGK 3 sort N3.3 For disk, generate DCBs, IOBs, DCB addresses AGK 3 sort For tape, read forward AGN 1 N3.1 OSCL: generate DCBs, IOBs, DCB addresses AGM 3 sort For tape, read forward AGN 1 N3.1 OSCL: generate DCBs and IOBs Contain messages for phase 1 assignment modules AMC 3 all Contain messages for phase 2 assignment modules AMC 3 all Sort fixed-length records, 'multiple' (not POLY) AOB 1 Sort fixed-length records, 'single' (not POLY) AOC 1 Sort var. records, 'multiple' (not POLY) AOC 1 Sort var. records, 'multiple' AOC 1 FOLY: sort fixed-length records, 'multiple' AOC 1 FOLY: sort fixed-length records, 'multiple' AOC 1 FOLY: sort var. r				
AGE 3 merge Open files AGG 2 N3.2 For tape, generate DCBs, IOBs, DCB addresses AGH 3 sort Open files (and call checkpoint module) AGI 1 N3.1 For disk, generate DCBs, IOBs, DCB addresses AGK 3 sort N3.2 For disk, generate DCBs, IOBs, DCB addresses AGK 3 sort N3.3 For disk, generate DCBs, IOBs, DCB addresses AGK 3 sort N3.3 For disk, generate DCBs, IOBs, DCB addresses AGK 3 sort For tape, read forward AGM 1 N3.1 OSCL: generate DCBs and IOBs Contain messages for phase 1 assignment modules Contain messages for phase 2 assignment modules Contain messages for phase 3 assignment modules AMC 3 all Contain messages for phase 3 assignment modules AMC 3 all Supply installed defaults Sort fixed-length records, 'multiple' (not POLY) AOB 1 Sort fixed-length records, 'single' (not POLY) AOC 1 Sort var. records, 'multiple' (not POLY) AOC 1 Sort var. records, 'single' (not POLY) AOC 1 POLY: sort fixed-length records, 'single' POLY: sort var. records, 'multiple' AOC 1 POLY: sort var. records, 'multiple' AOC 1 POLY: sort var. records, 'multiple' AOC 1 POLY: sort algorithm AOC 2 BALN (tape) merge algorithm Merge, 'multiple' Merge, '				
AGG 2 N3.2 For tape, generate DCBs, IOBs, DCB addresses Open files (and call checkpoint module) AGI 1 N3.1 For disk, generate DCBs, IOBs, DCB addresses AGI 2 N3.2 For disk, generate DCBs, IOBs, DCB addresses AGK 3 sort N3.3 For disk, generate DCBs, IOBs, DCB addresses For tape, read forward AGM 3 sort For tape, read forward AGM 1 N3.1 OSCL: generate DCBs and IOBs AMA 1 CONTAIN messages for phase 1 assignment modules AMB 2 Contain messages for phase 2 assignment modules AMC 3 all Contain messages for phase 3 assignment modules AMC Supply installed defaults Sort fixed-length records, 'multiple' (not POLY) AOB 1 Sort fixed-length records, 'single' (not POLY) AOC 1 Sort var. records, 'multiple' (not POLY) AOC 1 Sort var. records, 'single' (not POLY) AOC 1 POLY: sort fixed-length records, 'multiple' AOC 1 POLY: sort fixed-length records, 'multiple' AOC 1 POLY: sort var. records, 'multiple' AOC 2 BALN (dask) sort algorithm AOC 2 6 3 all AOC 2 BALN (tape) merge algorithm AOC 2 BALN (tape) merge algorithm AOC 2 POLY merge algorithm				
AGH 3 sort Open files (and call checkpoint module) AGI 1 N3.1 For disk, generate DCBs, IOBs, DCB addresses AGK 3 sort N3.3 For disk, generate DCBs, IOBs, DCB addresses AGK 3 sort N3.3 For disk, generate DCBs, IOBs, DCB addresses AGK 3 sort N3.3 For disk, generate DCBs, IOBs, DCB addresses AGK 3 sort N3.1 For disk, generate DCBs, IOBs, DCB addresses AGK 3 sort N3.3 For disk, generate DCBs, IOBs, DCB addresses AGK 3 sort N3.1 OSCL: generate DCBs and IOBs AMA 1 Contain messages for phase 1 assignment modules AMB 2 Contain messages for phase 2 assignment modules AMC 3 all Contain messages for phase 3 assignment modules AMC 3 all Contain messages for phase 3 assignment modules AMA 1 Sort fixed-length records, 'multiple' (not POLY) AOB 1 Sort fixed-length records, 'single' (not POLY) AOC 1 Sort var. records, 'multiple' (not POLY) AOC 1 Sort var. records, 'single' (not POLY) AOC 1 POLY: sort fixed-length records, 'multiple' POLY: sort fixed-length records, 'single' POLY: sort var. records, 'multiple' POLY: sort var. records, 'single' POLY: sort algorithm AON 1 BALN (disk) sort algorithm AON 0 (Opt) all D3.4 Generate equals routine OSCL sort algorithm AOR 2 BALN (tape) merge algorithm AOR 2 BALN (tape) merge algorithm POLY merge algorithm POLY merge algorithm	AGF			
AGH 3 sort Open files (and call checkpoint module) AGI 1 N3.1 For disk, generate DCBs, IOBs, DCB addresses AGK 3 sort N3.3 For disk, generate DCBs, IOBs, DCB addresses AGK 3 sort N3.3 For disk, generate DCBs, IOBs, DCB addresses AGK 3 sort N3.3 For disk, generate DCBs, IOBs, DCB addresses AGK 3 sort N3.1 For disk, generate DCBs, IOBs, DCB addresses AGK 3 sort N3.3 For disk, generate DCBs, IOBs, DCB addresses AGK 3 sort N3.1 OSCL: generate DCBs and IOBs AMA 1 Contain messages for phase 1 assignment modules AMB 2 Contain messages for phase 2 assignment modules AMC 3 all Contain messages for phase 3 assignment modules AMC 3 all Contain messages for phase 3 assignment modules AMA 1 Sort fixed-length records, 'multiple' (not POLY) AOB 1 Sort fixed-length records, 'single' (not POLY) AOC 1 Sort var. records, 'multiple' (not POLY) AOC 1 Sort var. records, 'single' (not POLY) AOC 1 POLY: sort fixed-length records, 'multiple' POLY: sort fixed-length records, 'single' POLY: sort var. records, 'multiple' POLY: sort var. records, 'single' POLY: sort algorithm AON 1 BALN (disk) sort algorithm AON 0 (Opt) all D3.4 Generate equals routine OSCL sort algorithm AOR 2 BALN (tape) merge algorithm AOR 2 BALN (tape) merge algorithm POLY merge algorithm POLY merge algorithm	AGG	2	N3.2	For tape, generate DCBs, IOBs, DCB addresses
AGJ 2 N3.2 For disk, generate DCBs, IOBs, DCB addresses AGK 3 sort N3.3 For disk, generate DCBs, IOBs, DCB addresses AGK 3 sort For tape, read forward AGM 3 sort For tape, read forward AGM 1 N3.1 OSCL: generate DCBs and IOBs AMA 1 Contain messages for phase 1 assignment modules AMB 2 Contain messages for phase 2 assignment modules AMC 3 all Contain messages for phase 3 assignment modules AMC 3 all Supply installed defaults AOA 1 Sort fixed-length records, 'multiple' (not POLY) AOB 1 Sort fixed-length records, 'single' (not POLY) AOC 1 Sort var. records, 'multiple' (not POLY) AOC 1 Sort var. records, 'single' (not POLY) AOC 1 Sort var. records, 'single' (not POLY) AOC 1 POLY: sort fixed-length records, 'single' POLY: sort fixed-length records, 'single' AOC 1 POLY: sort fixed-length records, 'single' AOC 1 POLY: sort var. records, 'multiple' AOC 1 POLY: sort var. records, 'single' AOC 1 POLY: sort algorithm AOC 1 POLY: sort algorithm AOC 0 (Opt) all D3.4 Generate equals routine AOC 2 BALN (tape) merge algorithm			•••	Open files (and call checkpoint module)
AGK 3 sort N3.3 For disk, generate DCBs, IOBs, DCB addresses AGL 2 For tape, read forward For tape, read forward For tape, read forward AGN 1 N3.1 OSCL: generate DCBs and IOBs Contain messages for phase 1 assignment modules Contain messages for phase 2 assignment modules AMC 3 all Contain messages for phase 3 assignment modules AMC 3 all Contain messages for phase 3 assignment modules AMI 0 Supply installed defaults Sort fixed-length records, 'multiple' (not POLY) AOB 1 Sort var. records, 'multiple' (not POLY) AOC 1 Sort var. records, 'multiple' (not POLY) AOC 1 Sort var. records, 'single' (not POLY) AOF 1 POLY: sort fixed-length records, 'single' POLY: sort var. records, 'multiple' AOF 1 POLY: sort var. records, 'multiple' POLY: sort var. records, 'multiple' AOG 1 POLY: sort var. records, 'single' POLY: sort algorithm BALN (tape) sort algorithm AON 1 OCCL sort algorithm AON 1 OSCL sort algorithm Merge, 'multiple' Merge, 'multiple' Merge, 'multiple' Merge, 'miltiple' Merge, 'miltip	AGI			For disk, generate DCBs, IOBs, DCB addresses
AGL 2 AGM 3 sort For tape, read forward For tape, read forward AGN 1 AGN 2 Contain messages for phase 1 assignment modules AGN 2 Contain messages for phase 2 assignment modules AGN 3 AGN 4 AGN 4 AGN 4 AGN 5 AGN 6 AGN 7 AGN		2		For disk, generate DCBs, IOBs, DCB addresses
AGM 3 sort For tape, read forward AGN 1 N3.1 OSCL: generate DCBs and IOBs AMA 1 Contain messages for phase 1 assignment modules AMB 2 Contain messages for phase 2 assignment modules AMC 3 all Contain messages for phase 3 assignment modules AMI 0 Supply installed defaults AOA 1 Sort fixed-length records, 'multiple' (not POLY) AOB 1 Sort fixed-length records, 'single' (not POLY) AOC 1 Sort var. records, 'multiple' (not POLY) AOC 1 Sort var. records, 'single' (not POLY) AOC 1 POLY: sort fixed-length records, 'multiple' AOF 1 POLY: sort fixed-length records, 'multiple' AOF 1 POLY: sort fixed-length records, 'single' AOG 1 POLY: sort fixed-length records, 'single' AOG 1 POLY: sort var. records, 'multiple' AOG 1 POLY: sort var. records, 'single' AOG 1 BALN (tape) sort algorithm AON 1 BALN (disk) sort algorithm AOM 0 (Opt) all D3.4 Generate equals routine Generate extract routine (part 1) OSCL sort algorithm AOP 2 6 3 all Merge, 'multiple' Merge, 'multiple' Merge, 'single' AOR 2 BALN (tape) merge algorithm POLY merge algorithm			из.3	
AMA 1 Contain messages for phase 1 assignment modules AMB 2 Contain messages for phase 2 assignment modules AMC 3 all Contain messages for phase 3 assignment modules AMI 0 Supply installed defaults AOA 1 Sort fixed-length records, 'multiple' (not POLY) AOB 1 Sort fixed-length records, 'single' (not POLY) AOC 1 Sort var. records, 'multiple' (not POLY) AOC 1 Sort var. records, 'single' (not POLY) AOC 1 POLY: sort fixed-length records, 'multiple' AOF 1 POLY: sort fixed-length records, 'single' AOF 1 POLY: sort fixed-length records, 'single' AOF 1 POLY: sort var. records, 'multiple' AOF 1 POLY: sort var. records, 'single' AOC 1 BALN (tape) sort algorithm AOC 1 BALN (disk) sort algorithm AOC 0 (Opt) all D3.4 Generate equals routine AOM 0 (Opt) all D3.4 Generate extract routine (part 1) AOC 2 BALN (tape) merge algorithm AOC 2 BALN (tape) merge algorithm BALN (tape) merge algorithm BALN (tape) merge algorithm BALN (tape) merge algorithm		2 3 sort	ε	For tape, read forward For tape, read forward
AMA 1 Contain messages for phase 1 assignment modules AMB 2 Contain messages for phase 2 assignment modules AMC 3 all Contain messages for phase 3 assignment modules AMI 0 Supply installed defaults AOA 1 Sort fixed-length records, 'multiple' (not POLY) AOB 1 Sort fixed-length records, 'single' (not POLY) AOC 1 Sort var. records, 'multiple' (not POLY) AOC 1 Sort var. records, 'single' (not POLY) AOC 1 POLY: sort fixed-length records, 'multiple' AOF 1 POLY: sort fixed-length records, 'single' AOF 1 POLY: sort fixed-length records, 'single' AOF 1 POLY: sort var. records, 'multiple' AOF 1 POLY: sort var. records, 'single' AOC 1 BALN (tape) sort algorithm AOC 1 BALN (disk) sort algorithm AOC 0 (Opt) all D3.4 Generate equals routine AOM 0 (Opt) all D3.4 Generate extract routine (part 1) AOC 2 BALN (tape) merge algorithm AOC 2 BALN (tape) merge algorithm BALN (tape) merge algorithm BALN (tape) merge algorithm BALN (tape) merge algorithm	3 (3)	4	x12 4	OCCI, conorate DCBs and TOBs
AMB 2 Contain messages for phase 2 assignment modules AMC 3 all Contain messages for phase 3 assignment modules AMI 0 Supply installed defaults AOA 1 Sort fixed-length records, 'multiple' (not POLY) AOB 1 Sort fixed-length records, 'single' (not POLY) AOC 1 Sort var. records, 'multiple' (not POLY) AOC 1 Sort var. records, 'single' (not POLY) AOE 1 POLY: sort fixed-length records, 'multiple' POLY: sort var. records, 'multiple' AOF 1 POLY: sort fixed-length records, 'single' AOG 1 POLY: sort var. records, 'multiple' POLY: sort var. records, 'multiple' AOI 1 POLY: sort var. records, 'single' BALN (tape) sort algorithm AOL 0 (Opt) all D3.4 Generate equals routine AOM 0 (Opt) all D3.4 Generate extract routine (part 1) AOR 2 BALN (tape) merge algorithm AOR 2 BALN (tape) merge algorithm POLY merge algorithm			NJ. 1	Contain messages for phase 1 assignment
AM1 0 Supply installed defaults AOA 1 Sort fixed-length records, 'multiple' (not POLY) AOB 1 Sort fixed-length records, 'single' (not POLY) AOC 1 Sort var. records, 'multiple' (not POLY) AOD 1 Sort var. records, 'single' (not POLY) AOE 1 POLY: sort fixed-length records, 'multiple' AOF 1 POLY: sort fixed-length records, 'single' AOG 1 POLY: sort var. records, 'multiple' AOH 1 POLY: sort var. records, 'multiple' AOH 1 POLY: sort var. records, 'single' AOI 1 BALN (tape) sort algorithm AOK 1 BALN (disk) sort algorithm AOL 0 (Opt) all D3.4 Generate equals routine AOM 0 (Opt) all D3.4 Generate extract routine (part 1) AON 1 AOP 2 6 3 all Merge, 'multiple' AOQ 2 8 3 all Merge, 'single' BALN (tape) merge algorithm AOR 2 BALN (tape) merge algorithm POLY merge algorithm	AMB	2		modules
AOA 1 Sort fixed-length records, 'multiple' (not POLY) AOB 1 Sort fixed-length records, 'single' (not POLY) AOC 1 Sort var. records, 'multiple' (not POLY) AOD 1 POLY: sort fixed-length records, 'multiple' AOF 1 POLY: sort fixed-length records, 'multiple' AOF 1 POLY: sort var. records, 'multiple' AOG 1 POLY: sort var. records, 'multiple' AOH 1 POLY: sort var. records, 'multiple' AOI 1 BALN (tape) sort algorithm AON 0 (Opt) all D3.4 Generate equals routine AOM 0 (Opt) all D3.4 Generate extract routine (part 1) AON 1 OSCL sort algorithm AOP 2 6 3 all Merge, 'multiple' AOR 2 BALN (tape) merge algorithm AOR 2 BALN (tape) merge algorithm AOS 2 BALN (tape) merge algorithm	AMC	3 all		
AOA 1 Sort fixed-length records, 'multiple' (not POLY) AOB 1 Sort fixed-length records, 'single' (not POLY) AOC 1 Sort var. records, 'multiple' (not POLY) AOD 1 POLY: sort fixed-length records, 'multiple' AOF 1 POLY: sort fixed-length records, 'multiple' AOF 1 POLY: sort var. records, 'multiple' AOG 1 POLY: sort var. records, 'multiple' AOH 1 POLY: sort var. records, 'multiple' AOI 1 BALN (tape) sort algorithm AON 0 (Opt) all D3.4 Generate equals routine AOM 0 (Opt) all D3.4 Generate extract routine (part 1) AON 1 OSCL sort algorithm AOP 2 6 3 all Merge, 'multiple' AOR 2 BALN (tape) merge algorithm AOR 2 BALN (tape) merge algorithm AOS 2 BALN (tape) merge algorithm	AM1	0		Supply installed defaults
AOC 1 Sort var. records, 'multiple' (not POLY) AOD 1 Sort var. records, 'single' (not POLY) AOE 1 POLY: sort fixed-length records, 'multiple' AOF 1 POLY: sort fixed-length records, 'single' AOG 1 POLY: sort var. records, 'multiple' AOG 1 POLY: sort var. records, 'multiple' AOI 1 POLY: sort var. records, 'single' AOI 1 BALN (tape) sort algorithm AOJ 1 POLY sort algorithm AOK 1 BALN (disk) sort algorithm AOL 0 (Opt) all D3.4 Generate equals routine AOM 0 (Opt) all D3.4 Generate extract routine (part 1) AON 1 OSCL sort algorithm AOP 2 & 3 all Merge, 'multiple' AOQ 2 & 3 all Merge, 'single' BALN (tape) merge algorithm AOR 2 BALN (tape) merge algorithm POLY merge algorithm		_		Sort fixed-length records, 'multiple'
AOD 1 Sort var. records, 'single' (not POLY) AOE 1 POLY: sort fixed-length records, 'multiple' AOF 1 POLY: sort fixed-length records, 'single' AOG 1 POLY: sort var. records, 'multiple' AOH 1 POLY: sort var. records, 'single' AOI 1 BALN (tape) sort algorithm AOJ 1 POLY sort algorithm AOK 1 BALN (disk) sort algorithm AOL 0 (Opt) all D3.4 Generate equals routine AOM 0 (Opt) all D3.4 Generate extract routine (part 1) AON 1 OSCL sort algorithm AOP 2 & 3 all Merge, 'multiple' AOQ 2 & 3 all Merge, 'single' BALN (tape) merge algorithm AOR 2 BALN (tape) merge algorithm POLY merge algorithm	AOB	1		(not POLY)
AOF 1 POLY: sort fixed-length records, 'single' AOG 1 POLY: sort var. records, 'multiple' AOH 1 POLY: sort var. records, 'single' AOI 1 BALN (tape) sort algorithm AOJ 1 POLY sort algorithm AOK 1 BALN (disk) sort algorithm AOK 1 BALN (disk) sort algorithm AOL 0 (Opt) all D3.4 Generate equals routine AOM 0 (Opt) all D3.4 Generate extract routine (part 1) AON 1 OSCL sort algorithm AOP 2 6 3 all Merge, 'multiple' AOQ 2 6 3 all Merge, 'single' BALN (tape) merge algorithm AOR 2 BALN (tape) merge algorithm POLY merge algorithm		-		Sort var. records, 'multiple' (not POLY) Sort var. records, 'single' (not POLY)
AOF 1 POLY: sort fixed-length records, 'single' AOG 1 POLY: sort var. records, 'multiple' AOH 1 POLY: sort var. records, 'single' AOI 1 BALN (tape) sort algorithm AOJ 1 POLY sort algorithm AOK 1 BALN (disk) sort algorithm AOK 1 BALN (disk) sort algorithm AOL 0 (Opt) all D3.4 Generate equals routine AOM 0 (Opt) all D3.4 Generate extract routine (part 1) AON 1 OSCL sort algorithm AOP 2 6 3 all Merge, 'multiple' AOQ 2 6 3 all Merge, 'single' BALN (tape) merge algorithm AOR 2 BALN (tape) merge algorithm POLY merge algorithm	AOE	1		POLY: sort fixed-length records. 'multiple'
AOG 1 AOH 1 POLY: sort var. records, 'multiple' POLY: sort var. records, 'single' AOI 1 BALN (tape) sort algorithm AOJ 1 POLY sort algorithm BALN (disk) sort algorithm AOK 1 BALN (disk) sort algorithm AOL 0 (Opt) all D3.4 Generate equals routine AOM 0 (Opt) all D3.4 Generate extract routine (part 1) AON 1 OSCL sort algorithm AOP 2 & 3 all Merge, 'multiple' AOQ 2 & 3 all Merge, 'single' BALN (tape) merge algorithm AOR 2 POLY merge algorithm		•		POLY: sort fixed-length records, 'single'
AOI 1 BALN (tape) sort algorithm AOJ 1 POLY sort algorithm AOK 1 BALN (disk) sort algorithm AOL 0 (Opt) all D3.4 Generate equals routine AOM 0 (Opt) all D3.4 Generate extract routine (part 1) AON 1 OSCL sort algorithm AOP 2 6 3 all Merge, "multiple" AOQ 2 6 3 all Merge, "single" AOR 2 BALN (tape) merge algorithm AOS 2 POLY merge algorithm		1		POLY: sort var. records, 'multiple'
AOJ 1 AOK 1 POLY sort algorithm BALN (disk) sort algorithm AOL 0 (Opt) all D3.4 Generate equals routine AOM 0 (Opt) all D3.4 Generate extract routine (part 1) AON 1 OSCL sort algorithm AOP 2 & 3 all Merge, 'multiple' AOQ 2 & 3 all Merge, 'single' AOR 2 BALN (tape) merge algorithm AOS 2 POLY merge algorithm		•		POLY: sort var. records, 'single'
AOK 1 BALN (disk) sort algorithm AOL 0 (Opt) all D3.4 Generate equals routine AOM 0 (Opt) all D3.4 Generate extract routine (part 1) AON 1 OSCL sort algorithm AOP 2 & 3 all Merge, "multiple" AOQ 2 & 3 all Merge, "single" AOR 2 BALN (tape) merge algorithm AOS 2 POLY merge algorithm				BALN (tape) sort algorithm
AOM 0 (Opt) all D3.4 Generate extract routine (part 1) AON 1 OSCL sort algorithm AOP 2 & 3 all Merge, "multiple" AOQ 2 & 3 all Merge, "single" AOR 2 BALN (tape) merge algorithm AOS 2 POLY merge algorithm				
AON 1 OSCL sort algorithm AOP 2 & 3 all Merge, "multiple" AOQ 2 & 3 all Merge, 'single' AOR 2 BALN (tape) merge algorithm AOS 2 POLY merge algorithm		0 (Opt) al	1 D3.4	Generate equals routine
AOP 2 & 3 all Merge, 'multiple' AOQ 2 & 3 all Merge, 'single' AOR 2 BALN (tape) merge algorithm AOS 2 POLY merge algorithm			1 D3.4	Generate extract routine (part 1)
AOQ 2 & 3 all Merge, 'single' AOR 2 BALN (tape) merge algorithm AOS 2 POLY merge algorithm				
AOS 2 POLY merge algorithm		2 & 3 all		Merge, 'single'
AOS 2 POLY merge algorithm	AOR	2		BALN (tape) merge algorithm
		2		POLY merge algorithm
	AOT			

	Used in MO Phase: Diag	Purpose
AOW	1	CRCX & OSCL: initialization for fixed-length records, 'multiple'
AOX	1	CRCX & OSCL: initialization for fixed-length records, 'single'
AOY	1	CRCX & OSCL: initialization for var. records, 'multiple'
AOZ	1	CRCX & OSCL: initialization for var. records, 'single'
AO1	0 (Opt) sort	Optimize disk unit assignment
AO2	0 (Opt) sort	Optimize tape unit assignment
AO4	0 (Opt) all D3.4	Generate extract routine (part 2)
A05	0 (Opt) all D3.4	Generate extract routine (part 3)
APA	1	Write tape
APB	1	Write disk
APC	1	Open files (and call checkpoint module); skip records on SORTIN
APD	2	Write tape
APE	_	Write disk
APF	3 merge N3.3	
APG	1 N3.1	(three, control blocks, etc.)
APH	2 N3.2	Allocate storage for I/O buffers, control blocks, etc.
API	3 all N3.3	Allocate storage for I/O buffers, control blocks, etc.
APJ	2	Open files (and call checkpoint module)
APK	3 sort N3.3	For tape: generate DCBs, IOBs, DCB addresses
APL	1 N3.1	CRCX & OSCL: allocate storage for I/O buffers, RSA, control blocks, etc.
AP1	1	Specify area for DCB list for OPEN
AP2	2	Specify area for DCB list for OPEN
AP3	3	Specify area for DCB list for OPEN
BGB	0 (Def) sort D3.4	CRCX: calculate B and G
CHK DM4	1,2,3 sort All	Checkpoint routine for sorting applications Hexadecimal and decimal conversion
EX1	1	Act as link between user routines and program exits.
EX2	2	Routines which need linkage editing are linkage edited together with these modules:
EX3	3 all	phase 1 routines with EX1, and so on.
RBA	1 N4.11	
RBB	1 N4.11	
RBC	1 N4.11	Block fixed-length records with link to multiple move
RBE	1 N4.11	
RBF		Move var. records
RBG	2 N3.2	Block or deblock fixed-length records with in-line move (no exits)

RBH 2 N3.2 Block or deblock fixed-length records with link to multiple move (no exits) RBJ 2 N3.2 Block or deblock var. records with move (no exits) RBJ 2 N3.2 Block or deblock var. records with move (no exits) RBK 2 N3.2 Block or deblock var. records (exits used) RBK 2 N3.2 Block or deblock var. records (exits used) RBK 3 all N3.3 Block fixed-length records with in-line move (no exits) RBM 3 all N3.3 Block fixed-length records (exits used) RBM 3 all N3.3 Block var. records (exits used) RBM 3 all N3.3 Block var. records (exits used) RBM 3 all N3.3 Block var. records (exits used) RBP 3 all N3.3 Block var. records (exits used) RBP 3 all N3.3 Block fixed-length records with link to multiple move RBT 2 N3.2 CRCX & OSCL: block/deblock fixed-length records > 256 bytes (no exits) RBU 2 N3.2 CRCX & OSCL: block/deblock var. records (no exits) RBW 2 N3.2 CRCX & OSCL: block/deblock var. records (no exits) RBW 2 N3.2 CRCX & OSCL: block/deblock var. records (exits used) RBW 2 N3.2 CRCX & OSCL: block/deblock var. records (exits used) RBW 2 N3.2 CRCX & OSCL: block/deblock var. records (exits used) RBW 2 N3.2 CRCX & OSCL: block/deblock var. records (exits used) RBW 2 N3.2 CRCX & OSCL: block/deblock var. records (exits used) RBW 2 N3.2 CRCX & OSCL: block/deblock var. records (exits used) RBW 2 N3.2 CRCX & OSCL: block fixed-length records (exits used) RBW 1 N4.11 CRCX & OSCL: block fixed-length records (exits used) RBW 2 N3.2 CRCX & OSCL: block fixed-length records (exits used) RBW 3 SCAN BODS Control Statement Identify input, output and work data sets and their characteristics RCJ 0(Def) all D3.4 SCAN BODS Control Statement Identify input, output and work data sets and their characteristics RCJ 0(Def) all D3.4 SCAN BODS Control Statements Calculate bin sizes RCC 0(Def) all D3.4 SCAN BODS Control Statements Calculate B and G for tape sort and choose (exits used) RCS 0(Def) all D3.4 Optimal technique; interpret MODS statements Calculate B and G for tape sort and choose Calculate B and G for tape sort a		Used in Phase:	Diag	
RBI 2 N3.2 Block or deblock var. records with move (no exits) RBK 2 N3.2 Block or deblock var. records (exits used) RBK 2 N3.3 Block fixed-length records (exits used) RBK 3 all N3.3 Block fixed-length records (exits used) RBM 3 all N3.3 Block fixed-length records (exits used) RBM 3 all N3.3 Block var. records (exits used) RBM 3 all N3.3 Block var. records (exits used) RBM 3 all N3.3 Block var. records (exits used) RBM 3 all N3.3 Block var. records (exits used) RBM 3 all N3.3 Block fixed-length records with link to multiple move RBT 2 N3.2 CRCX & OSCL: block/deblock fixed-length records up to 256 bytes (no exits) RBW 2 N3.2 CRCX & OSCL: block/deblock fixed-length records (exits used) RBW 2 N3.2 CRCX & OSCL: block/deblock fixed-length records (exits used) RBW 2 N3.2 CRCX & OSCL: block/deblock var. records (exits used) RBW 2 N3.2 CRCX & OSCL: block/deblock var. records (exits used) RBW 2 N3.2 CRCX & OSCL: block/deblock var. records (exits used) RBW 2 N3.2 CRCX & OSCL: block/deblock var. records (exits used) RBW 1 N4.11 CRCX & OSCL: block fixed-length records (exits used) RBY 1 N4.11 CRCX & OSCL: block fixed-length records (exits used) RBY 1 N4.11 CRCX & OSCL: block fixed-length records (exits used) RBY 1 N4.11 CRCX & OSCL: block fixed-length records (exits used) RBY 1 N4.11 CRCX & OSCL: block fixed-length records (exits used) RBY 1 N4.11 CRCX & OSCL: block fixed-length records (exits used) RBY 1 N4.11 CRCX & OSCL: block fixed-length records (exits used) RBY 1 N4.11 CRCX & OSCL: block fixed-length records (exits used) RBY 1 N4.11 CRCX & OSCL: block fixed-length records (exits used) RBY 1 N4.11 CRCX & OSCL: block fixed-length records (exits used) RBY 1 N4.11 CRCX & OSCL: block fixed-length records (exits used) RBY 1 N4.11 CRCX & OSCL: block fixed-length records (exits used) RBY 1 N4.11 CRCX & OSCL: block fixed-length records (exits used) RBY 1 N4.11 CRCX & OSCL: block fixed-length records (exits used) RBY 1 N4.11 CRCX & OSCL: block fixed-length records (exits used) RBY 1 N4.11 CRCX &		_		
RBJ 2 N3.2 Block or deblock fixed-length records (exits used) RBL 3 all N3.3 Block fixed-length records (exits used) RBM 3 all N3.3 Block fixed-length records (exits used) RBM 3 all N3.3 Block fixed-length records (exits used) RBM 3 all N3.3 Block var. records (exits used) RBP 3 all N3.3 Block var. records (exits used) RBP 3 all N3.3 Block var. records (exits used) RBP 3 all N3.3 Block var. records (exits used) RBP 3 all N3.3 Block var. records (exits used) RBP 3 all N3.3 Block var. records (exits used) RBP 2 N3.2 CRCX & OSCL: block/deblock fixed-length records up to 256 bytes (no exits) RBU 2 N3.2 CRCX & OSCL: block/deblock fixed-length records (exits used) RBW 2 N3.2 CRCX & OSCL: block/deblock var. records (no exits) RBW 2 N3.2 CRCX & OSCL: block/deblock var. records (exits used) RBX 2 N3.2 CRCX & OSCL: block/deblock var. records (exits used) RBX 2 N3.2 CRCX & OSCL: block/deblock var. records (exits used) RBX 1 N4.11 CRCX & OSCL: block/deblock var. records (exits used) RBX 2 N3.2 CRCX & OSCL: block fixed-length records up to 256 bytes RBZ 1 N4.11 CRCX & OSCL: block fixed-length records var. records (exits used) RBZ 1 N4.11 CRCX & OSCL: block fixed-length records var. records (exits used) RBX 2 N3.2 CRCX & OSCL: block fixed-length records var. records (exits used) RBX 2 N3.2 CRCX & OSCL: block fixed-length records var. records (exits used) RBX 3 N3.2 CRCX & OSCL: block fixed-length records var. records (exits used) RBY 1 N4.11 CRCX & OSCL: block fixed-length records var. records (exits used) RBY 2 N3.2 CRCX & OSCL: block fixed-length records var. records (exits used) RBY 3 N3.2 CRCX & OSCL: block fixed-length records var. records (exits used) RBY 4 N3.2 CRCX & OSCL: block fixed-length records var. records (exits used) RBY 5 N3.2 CRCX & OSCL: block fixed-length records var. records (exits used) RBY 6 N3.2 CRCX & OSCL: block fixed-length records var. records (exits used) RBY 7 N3.2 CRCX & OSCL: block fixed-length records var. records (exits used) RBY 8 N3.2 CRCX & OSCL: block fixed-	RBH			link to multiple move (no exits)
REM 2 N3.2 Block or deblock fixed-length records (exits used) REM 3 all N3.3 Block fixed-length records with in-line move (no exits) REM 3 all N3.3 Block fixed-length records (exits used) REM 3 all N3.3 Block var. records (exits used) REM 3 all N3.3 Block var. records (exits used) REM 3 all N3.3 Block var. records (exits used) REM 3 all N3.3 Block var. records (exits used) REM 3 all N3.3 Block var. records (exits used) REM 3 all N3.3 Block var. records (exits used) REM 2 N3.2 CRCX & OSCL: block/deblock fixed-length records up to 256 bytes (no exits) REM 2 N3.2 CRCX & OSCL: block/deblock fixed-length records var. records (no exits) REM 2 N3.2 CRCX & OSCL: block/deblock fixed-length records (exits used) REM 2 N3.2 CRCX & OSCL: block/deblock var. records (exits used) REM 2 N3.2 CRCX & OSCL: block/deblock var. records (exits used) REM 2 N3.2 CRCX & OSCL: block/deblock var. records (exits used) REM 2 N3.2 CRCX & OSCL: block/deblock var. records (exits used) REM 2 N3.2 CRCX & OSCL: block/deblock var. records (exits used) REM 3 all but 0 (Def) CRCX & OSCL: block/deblock var. records (exits used) REM 4 N4.11 CRCX & OSCL: block/deblock fixed-length records var. var. var. var. var. var. var. var.	RBI	2	N3.2	
REK 2 N3.2 Block or deblock var. records (exits used) REL 3 all N3.3 Block fixed-length records with in-line move (no exits) REM 3 all N3.3 Block var. records (exits used) REM 3 all N3.3 Block var. records (exits used) REB 3 all N3.3 Block var. records (exits used) REB 3 all N3.3 Block var. records (exits used) REB 3 all N3.3 Block var. records (exits used) REB 3 all N3.3 Block var. records (exits used) REB 3 all N3.3 Block var. records (exits used) REB 2 N3.2 CRCX & OSCL: block/deblock fixed-length records up to 256 bytes (no exits) REB 2 N3.2 CRCX & OSCL: block/deblock fixed-length records (exits) REB 2 N3.2 CRCX & OSCL: block/deblock var. records (exits used) REB 2 N3.2 CRCX & OSCL: block/deblock fixed-length records (exits used) REB 2 N3.2 CRCX & OSCL: block/deblock var. records (exits used) REB 2 N3.2 CRCX & OSCL: block/deblock var. records (exits used) REB 2 N3.2 CRCX & OSCL: block/deblock var. records (exits used) REB 3 N4.11 CRCX & OSCL: block/deblock var. records (exits used) REB 1 N4.11 CRCX & OSCL: block fixed-length records var. var. var. var. var. var. var. var.	RBJ	2	N3.2	Block or deblock fixed-length records
RBM 3 all N3.3 Block fixed-length records (exits used) RBN 3 all N3.3 Block var. records (no exits) RBD 3 all N3.3 Block var. records (exits used) RBB 3 all N3.3 Block var. records (exits used) RBB 3 all N3.3 Block var. records (exits used) RBB 3 all N3.3 Block var. records (exits used) RBB 3 all N3.3 Block fixed-length records with link to multiple move RBT 2 N3.2 CRCX & OSCL: block/deblock fixed-length records vp to 256 bytes (no exits) RBU 2 N3.2 CRCX & OSCL: block/deblock fixed-length records > 256 bytes (no exits) CRCX & OSCL: block/deblock fixed-length records (exits used) RBW 2 N3.2 CRCX & OSCL: block/deblock fixed-length records (exits used) RBX 2 N3.2 CRCX & OSCL: block/deblock var. records (exits used) RBX 1 N4.11 CRCX & OSCL: block fixed-length records (exits used) RBZ 1 N4.11 CRCX & OSCL: block fixed-length records up to 256 bytes RCA All but 0 (Def) RCB O (Def) all O3.4 Scan MODS control statement RCCI 0 (Def) all Gall Marge-only optimization RCCI 0 (Def) all D3.4 System interface RCCI 0 (Def) all System interface controlling phase 0 (Def) RCC 0 (Def) all System interface controlling phase 0 (Def) RCC 0 (Def) all System interface control statements RCCI 0 (Def) sort D3.4 Calculate bin sizes RCCI 0 (Def) all System interface control statements RCCI 0 (Def) all System interface control statements RCCI 0 (Def) all System interface control statements RCCI 0 (Def) all Calculate B and G for tape sort and choose RCCI 0 (Def) all Contain messages for phase 0 RCCI 0 (Def) all Contain messages for phase 0 RCCI 0 (Def) all Contain messages for MODS card errors RCCI 0 (Def) all Contain messages for MODS card errors RCCI 0 (Def) all Contain messages for modes card errors RCCI 0 (Def) all Contain messages for modes card errors RCCI 0 (Def) all Contain messages for modes card errors RCCI 0 (Def) all Contain messages for modes card errors RCCI 0 (Def) all Contain messages for modes card errors RCCI 0 (Def) all Contain messages for mod	RBK	2	N3.2	Block or deblock var. records (exits used)
RBM 3 all N3.3 Block fixed-length records (exits used) RBO 3 all N3.3 Block var. records (no exits) RBO 3 all N3.3 Block var. records (exits used) RBF 3 all N3.3 Block var. records (exits used) RBF 3 all N3.3 Block var. records (exits used) RBF 2 N3.2 CRCX & OSCL: block/deblock fixed-length records up to 256 bytes (no exits) RBU 2 N3.2 CRCX & OSCL: block/deblock fixed-length records 2556 bytes (no exits) RBW 2 N3.2 CRCX & OSCL: block/deblock var. records (no exits) RBW 2 N3.2 CRCX & OSCL: block/deblock fixed-length records (exits used) RBW 2 N3.2 CRCX & OSCL: block/deblock var. records (exits used) RBW 1 N4.11 CRCX & OSCL: block/deblock var. records (exits used) RBZ 1 N4.11 CRCX & OSCL: block fixed-length records up to 256 bytes RBZ 1 N4.11 CRCX & OSCL: block fixed-length records 256 bytes RCA All but 0 (Def) Specify PPI area RCB All but 0 (Def) System interface RCD 0 (Def) all D3.4 Scan MODS control statement Identify input, output and work data sets and their characteristics RCJ 0 (Opt) sort Check disk capacity RCM 0 (Def) all D3.4 Calculate bin sizes RCO 0 (Def) all D3.4 Calculate bin sizes RCO 0 (Def) all D3.4 Calculate bin sizes RCO 0 (Def) all D3.4 Calculate B and G for tape sort and choose (Def) (Def) all Calculate B and G for tape sort and choose (Def) (Def) all Calculate B and G for tape sort and choose (Def) (Def) all Control of phase 0 (Def) (Def) all Control of phase 0 (Def) (Def) all Control of phase 0 (Det) (Def) (Def) all Control of phase for control calculate Control of phase 0 (Det) (Def) (Def) all Calculation for extract	RBL	3 all	из.3	
RBN 3 all N3.3 Block var. records (no exits) RBP 3 all N3.3 Block var. records (exits used) RBP 3 all N3.3 Block fixed-length records with link to multiple move RBT 2 N3.2 CRCX & OSCL: block/deblock fixed-length records up to 256 bytes (no exits) RBU 2 N3.2 CRCX & OSCL: block/deblock fixed-length records >256 bytes (no exits) RBW 2 N3.2 CRCX & OSCL: block/deblock fixed-length records (exits used) CRCX & OSCL: block/deblock fixed-length records (exits used) CRCX & OSCL: block/deblock fixed-length records (exits used) CRCX & OSCL: block/deblock var. records (exits used) RBX 1 N4.11 CRCX & OSCL: block/deblock var. records (exits used) RBZ 1 N4.11 CRCX & OSCL: block/deblock var. records (exits used) RBZ 1 N4.11 CRCX & OSCL: block fixed-length records up to 256 bytes RBZ 1 N4.11 CRCX & OSCL: block fixed-length records var. records (exits used) RBZ 1 N4.11 CRCX & OSCL: block fixed-length records var. records (exits used) RBZ 1 N4.11 CRCX & OSCL: block fixed-length records var. records (exits used) RBZ 1 N4.11 CRCX & OSCL: block fixed-length records var. records (exits used) RBZ 1 N4.11 CRCX & OSCL: block fixed-length records var. records (exits used) RBZ 1 N4.11 CRCX & OSCL: block fixed-length records var. records va	R RM	3 =11	พร ร	Block fived-length records (evits used)
RBD 3 all N3.3 Block var. records (exits used) RBP 3 all N3.3 Block fixed-length records with link to multiple move RBT 2 N3.2 CRCX & OSCL: block/deblock fixed-length records up to 256 bytes (no exits) RBU 2 N3.2 CRCX & OSCL: block/deblock fixed-length records > 256 bytes (no exits) RBV 2 N3.2 CRCX & OSCL: block/deblock var. records (no exits) RBW 2 N3.2 CRCX & OSCL: block/deblock var. records (no exits) RBW 2 N3.2 CRCX & OSCL: block/deblock var. records (exits used) RBX 2 N3.2 CRCX & OSCL: block/deblock var. records (exits used) RBX 1 N4.11 CRCX & OSCL: block fixed-length records (exits used) RBY 1 N4.11 CRCX & OSCL: block fixed-length records up to 256 bytes RBZ 1 N4.11 CRCX & OSCL: block fixed-length records > 256 bytes RBZ 1 N4.11 CRCX & OSCL: block fixed-length records 256 bytes 256 bytes 256 bytes 256 by		2 all		
RBF 3 all N3.3 Block fixed-length records with link to multiple move RBT 2 N3.2 CRCx & OSCL: block/deblock fixed-length records up to 256 bytes (no exits) RBU 2 N3.2 CRCx & OSCL: block/deblock rised-length records > 256 bytes (no exits) RBV 2 N3.2 CRCx & OSCL: block/deblock var. records (no exits) RBW 2 N3.2 CRCx & OSCL: block/deblock fixed-length records (exits used) RBX 2 N3.2 CRCx & OSCL: block/deblock var. records (exits used) RBX 1 N4.11 CRCx & OSCL: block/deblock var. records (exits used) RBZ 1 N4.11 CRCx & OSCL: block fixed-length records up to 256 bytes RCA All but 0 (Def) Specify PPI area RCB All but 0 (Def) System interface RCD 0 (Def) all D3.4 Scan MODS control statement RCI 0 (Def) all D3.4 System interface RCJ 0 (Opt) sort Check disk capacity RCK 0 (Opt) sort D3.4 BALN (disk): optimize B and G RCC 0 (Def) all D3.4 System interface RCC 0 (Def) all D3.4 System interface RCC 0 (Def) all System interface RCC 0 (Def) all D3.4 Calculate bin sizes RCC 0 (Def) all System interface controlling phase 0 (Def) (and linkage editor) RCC 0 (Def) all Specify input area for control statements RCC 0 (Def) sort D3.4 Calculate B and G for tape sort and choose RCC 0 (Def) all Specify input area for control statements RCC 0 (Def) all Contain error messages for phase 0 RCC 1,2,3 all CCC Contain error messages for phase 0 RCC 0 (Def) all Contain messages for MODS card errors RCC 0 (Def) all Control of phase 0 (Opt) RCC 0 (Def) all D3.4 Move CPI information to PPI RCC 0 (Def) all D3.4 Move CPI information to PPI RCC 0 (Def) all Calculation for extract		2 gTT	N3.3	Block var. records (no exits)
RBT 2 N3.2 CRCX & OSCL: block/deblock fixed-length records up to 256 bytes (no exits) RBU 2 N3.2 CRCX & OSCL: block/deblock fixed-length records > 256 bytes (no exits) RBV 2 N3.2 CRCX & OSCL: block/deblock var. records (no exits) RBW 2 N3.2 CRCX & OSCL: block/deblock fixed-length records (exits used) RBX 2 N3.2 CRCX & OSCL: block/deblock var. records (exits used) RBX 1 N4.11 CRCX & OSCL: block/deblock var. records (exits used) RBY 1 N4.11 CRCX & OSCL: block fixed-length records (exits used) RBZ 1 N4.11 CRCX & OSCL: block fixed-length records var. records (exits used) RBZ 1 N4.11 CRCX & OSCL: block fixed-length records > 256 bytes RCA All but 0 (Def) Specify PPI area System interface RCB All but 0 (Def) System interface RCB All but 0 (Def) System interface RCCI 0 (Opt) sort Industry input, output and work data sets and their characteristics RCCI 0 (Opt) sort D3.4 Merge-only optimize B and G RCCI 0 (Def) all D3.4 System interface RCCI 0 (Def) all D3.4 System interface RCCI 0 (Def) all D3.4 List exit routines to be linkage edited Specify input area for control statement Calculate B and G for tape sort and choose RCC D3.4 Optimal technique; interpret MODS statement Free storage between phases RCCI 1,2,3 all CRCY 1,2,3 all CRCY 0 (Def) all Contain messages for moDS card errors RCCI 0 (Opt) all D3.4 Move CPI information to PPI Calculation for extract		3 all	E.EM	Block var. records (exits used)
records up to 256 bytes (no exits) RBU 2 N3.2 CRCX & OSCL: block/deblock fixed-length records >256 bytes (no exits) RBV 2 N3.2 CRCX & OSCL: block/deblock var. records (no exits) RBW 2 N3.2 CRCX & OSCL: block/deblock fixed-length records (exits used) RBX 2 N3.2 CRCX & OSCL: block/deblock var. records (exits used) RBX 1 N4.11 CRCX & OSCL: block fixed-length records (exits used) RBZ 1 N4.11 CRCX & OSCL: block fixed-length records up to 256 bytes RBZ 1 N4.11 CRCX & OSCL: block fixed-length records var. records (exits used) RBZ 1 N4.11 CRCX & OSCL: block fixed-length records var. records (exits used) RBZ 1 N4.11 CRCX & OSCL: block fixed-length records var. records (exits used) RBZ 1 N4.11 CRCX & OSCL: block fixed-length records var. records var. records (exits used) RBZ 1 N4.11 CRCX & OSCL: block fixed-length records var. records var. records var. records (exits used) RBZ 1 N4.11 CRCX & OSCL: block fixed-length records var. records v	RBP	3 all	из.3	
RBU 2 N3.2 CRCX & OSCL: block/deblock fixed-length records > 256 bytes (no exits) RBW 2 N3.2 CRCX & OSCL: block/deblock var. records (no exits) RBW 2 N3.2 CRCX & OSCL: block/deblock fixed-length records (exits used) RBX 2 N3.2 CRCX & OSCL: block/deblock var. records (exits used) RBX 2 N3.2 CRCX & OSCL: block fixed-length records (exits used) RBX 1 N4.11 CRCX & OSCL: block fixed-length records up to 256 bytes RBZ 1 N4.11 CRCX & OSCL: block fixed-length records > 256 bytes RBZ 1 N4.11 CRCX & OSCL: block fixed-length records > 256 bytes RCA All but 0 (Def) Specify PPI area RCB All but 0 (Def) System interface RCD 0 (Def) all D3.4 System interface RCD 0 (Def) all D3.4 System interface RCC 0 (Def) sort D3.4 BALN (disk): optimize B and G RCC 0 (Def) all D3.4 System interface RCM 0 (Def) all System interface controlling phase 0 (Def) RCC 0 (Def) all System interface control statements RCC 0 (Def) all System interface control statements RCC 0 (Def) sort D3.4 Calculate bin sizes RCC 0 (Def) all System interface control statements RCC 0 (Def) all System interface control statements RCC 0 (Def) all System interface control statements RCC 0 (Def) all System interface for control statements RCC 0 (Def) all Contain error messages for phase 0 System interface for processing records Contain error messages for MODS card errors Contain messages for MODS card errors Contain messages for MODS card errors Contain for extract	RBT	2	N3.2	
RBW 2 N3.2 CRCX & OSCL: block/deblock var. records (no exits) RBW 2 N3.2 CRCX & OSCL: block/deblock fixed-length records (exits used) RBX 2 N3.2 CRCX & OSCL: block/deblock var. records (exits used) RBX 1 N4.11 CRCX & OSCL: block fixed-length records up to 256 bytes RBZ 1 N4.11 CRCX & OSCL: block fixed-length records up to 256 bytes RCA All but 0 (Def) Specify PPI area RCB All but 0 (Def) System interface RCD 0 (Def) all D3.4 Scan MODS control statement RCI 0 (Opt) sort Identify input, output and work data sets and their characteristics RCJ 0 (Opt) sort Check disk capacity RCK 0 (Opt) sort D3.4 BALN (disk): optimize B and G RCM 0 (Def) all D3.4 System interface RCM 0 (Def) all D3.4 System interface RCM 0 (Def) all D3.4 System interface RCM 0 (Def) all System interface RCO 0 (Def) all System interface RCO 0 (Def) all System interface controlling phase 0 (Def) (and linkage editor) RCF 0 (Def) all Specify input area for control statements RCC 0 (Def) all Specify input area for control statements RCC 0 (Def) all Calculate B and G for tape sort and choose RCS D3.4 Optimal technique; interpret MODS statement RCC 0 (Def) all System interface for processing records RCC 1,2,3 all System interface for processing records RCC 1,2,3 all System interface for processing records RCC 0 (Def) all Contain messages for MODS card errors RCC 0 (Opt) all D3.4 Move CPI information to PPI RCC 0 (Def) all D3.4 Move CPI information to PPI RCC 0 (Def) all Calculation for extract	RBU	2	N3.2	CRCX & OSCL: block/deblock fixed-length
RBW 2 N3.2 CRCX & OSCL: block/deblock fixed-length records (exits used) RBX 2 N3.2 CRCX & OSCL: block/deblock var. records (exits used) RBY 1 N4.11 CRCX & OSCL: block fixed-length records up to 256 bytes RBZ 1 N4.11 CRCX & OSCL: block fixed-length records var. records var. records (exits used) RBZ 1 N4.11 CRCX & OSCL: block fixed-length records var.	RBV	2	N3.2	CRCX & OSCL: block/deblock var. records
REY 1 N4.11 CRCX & OSCL: block fixed-length records up to 256 bytes REZ 1 N4.11 CRCX & OSCL: block fixed-length records up to 256 bytes REZ 1 N4.11 CRCX & OSCL: block fixed-length records vp to 256 bytes RCA All but 0 (Def) Specify PPI area RCB All but 0 (Def) System interface RCD 0 (Def) all D3.4 Scan MODS control statement RCI 0 (Def) all Identify input, output and work data sets and their characteristics RCJ 0 (Opt) sort Check disk capacity RCK 0 (Opt) sort D3.4 BALN (disk): optimize B and G RCM 0 (Def) all D3.4 System interface RCM 0 (Def) sort D3.4 Calculate bin sizes RCO 0 (Def) all System interface controlling phase 0 (Def) (and linkage editor) RCP 0 (Def) all D3.4 Calculate B and G for tape sort and choose RCS D3.4 Optimal technique; interpret MODS statements RCC 0 (Def) all Control of phase 0 (Opt) RCC 1,2,3 all Contain error messages for phase 0 RCX 0 (Def) all Control of phase 0 (Opt) RCC 0 (Opt) all D3.4 Move CPI information to PPI RCC 0 (Def) all D3.4 Move CPI information to PPI RCC 0 (Def) all Control of calculation for extract	RBW	2	N3.2	CRCX & OSCL: block/deblock fixed-length
REZ 1 N4.11 CRCX & OSCL: block fixed-length records >256 bytes RCA All but 0 (Def) Specify PPI area RCB All but 0 (Def) System interface RCD 0 (Def) all D3.4 Scan MODS control statement RCI 0 (Def) all Identify input, output and work data sets and their characteristics RCJ 0 (Opt) sort Check disk capacity RCK 0 (Opt) sort D3.4 BALN (disk): optimize B and G RCL 0 (Def) merge D3.4 Merge-only optimization RCM 0 (Def) all D3.4 System interface RCN 0 (Def) sort D3.4 Calculate bin sizes RCO 0 (Def) all System interface controlling phase 0 (Def) (and linkage editor) RCP 0 (Def) all Specify input area for control statements RCR 0 (Def) sort D3.4 Calculate B and G for tape sort and choose RCS D3.4 Optimal technique; interpret MODS statement RCT 1,2,3 all Free storage between phases RCU 0 (Def+Opt) all Contain error messages for phase 0 RCX 1,2,3 all System interface for processing records RCX 0 (Def) all Contain messages for MODS card errors RCX 0 (Def) all D3.4 Move CPI information to PPI RC2 0 (Def) all D3.4 Move CPI information to PPI RC2 0 (Def) all Calculation for extract	RBX	2	N3.2	CRCX & OSCL: block/deblock var. records
REZ 1 N4.11 CRCX & OSCL: block fixed-length records >256 bytes RCA All but 0 (Def) Specify PPI area RCB All but 0 (Def) System interface RCD 0 (Def) all D3.4 Scan MODS control statement RCI 0 (Def) all Identify input, output and work data sets and their characteristics RCJ 0 (Opt) sort Check disk capacity RCK 0 (Opt) sort D3.4 BALN (disk): optimize B and G RCL 0 (Def) merge D3.4 Merge-only optimization RCM 0 (Def) all D3.4 System interface RCN 0 (Def) sort D3.4 Calculate bin sizes RCO 0 (Def) all System interface controlling phase 0 (Def) (and linkage editor) RCP 0 (Def) all System interface control statements RCC 0 (Def) all Specify input area for control statements RCR 0 (Def) sort D3.4 Calculate B and G for tape sort and choose RCS D3.4 Optimal technique; interpret MODS statements RCC 1,2,3 all System interface for processing records RCC 1,2,3 all System interface for processing records RCC 0 (Def) all Contain messages for MODS card errors RCC 0 (Opt) all D3.4 Move CPI information to PPI RCC 0 (Def) all D3.4 Move CPI information to PPI RCC 0 (Def) all Calculation for extract	RBY	1	N4.11	
RCB All but 0 (Def) System interface RCD 0 (Def) all D3.4 Scan MODS control statement RCI 0 (Def) all Identify input, output and work data sets and their characteristics RCJ 0 (Opt) sort Check disk capacity RCK 0 (Opt) sort D3.4 BALN (disk): optimize B and G RCL 0 (Def) merge D3.4 Merge-only optimization RCM 0 (Def) all D3.4 System interface RCN 0 (Def) sort D3.4 Calculate bin sizes RCO 0 (Def) all System interface controlling phase 0 (Def) (and linkage editor) RCP 0 (Def) all D3.4 List exit routines to be linkage edited RCQ 0 (Def) all Specify input area for control statements RCR 0 (Def) sort D3.4 Calculate B and G for tape sort and choose RCS D3.4 Optimal technique; interpret MODS statement RCT 1,2,3 all System interface for processing records RCV 1,2,3 all System interface for processing records RCX 0 (Def) all Contain messages for MODS card errors RCZ 0 (Opt) all D3.4 Move CPI information to PPI RC2 0 (Def) all Calculation for extract	RBZ	1	N4.11	CRCX & OSCL: block fixed-length records
RCB All but 0 (Def) System interface RCD 0 (Def) all D3.4 Scan MODS control statement RCI 0 (Def) all Identify input, output and work data sets and their characteristics RCJ 0 (Opt) sort Check disk capacity RCK 0 (Opt) sort D3.4 BALN (disk): optimize B and G RCL 0 (Def) merge D3.4 Merge-only optimization RCM 0 (Def) all D3.4 System interface RCN 0 (Def) sort D3.4 Calculate bin sizes RCO 0 (Def) all System interface controlling phase 0 (Def) (and linkage editor) RCP 0 (Def) all D3.4 List exit routines to be linkage edited RCQ 0 (Def) all Specify input area for control statements RCR 0 (Def) sort D3.4 Calculate B and G for tape sort and choose RCS D3.4 Optimal technique; interpret MODS statement RCT 1,2,3 all System interface for processing records RCV 1,2,3 all System interface for processing records RCX 0 (Def) all Contain messages for MODS card errors RCZ 0 (Opt) all D3.4 Move CPI information to PPI RC2 0 (Def) all Calculation for extract	RCA	All but 0 (D	ef)	Specify PPI area
RCI 0 (Def) all Identify input, output and work data sets and their characteristics RCJ 0 (Opt) sort Check disk capacity RCK 0 (Opt) sort D3.4 BALN (disk): optimize B and G RCL 0 (Def) merge D3.4 Merge-only optimization RCM 0 (Def) all D3.4 System interface RCN 0 (Def) sort D3.4 Calculate bin sizes RCO 0 (Def) all System interface controlling phase 0 (Def) (and linkage editor) RCP 0 (Def) all D3.4 List exit routines to be linkage edited RCQ 0 (Def) all Specify input area for control statements RCR 0 (Def) sort D3.4 Calculate B and G for tape sort and choose RCS D3.4 Optimal technique; interpret MODS statement RCT 1,2,3 all Free storage between phases RCU 0 (Def+Opt) all Contain error messages for phase 0 RCV 1,2,3 all System interface for processing records RCX 0 (Def) all Contain messages for MODS card errors RCZ 0 (Opt) all D3.4 Move CPI information to PPI RC2 0 (Def) all Calculation for extract	RCB	All but 0 (D	ef)	System interface
RCI 0 (Def) all Identify input, output and work data sets and their characteristics RCJ 0 (Opt) sort Check disk capacity RCK 0 (Opt) sort D3.4 BALN (disk): optimize B and G RCL 0 (Def) merge D3.4 Merge-only optimization RCM 0 (Def) all D3.4 System interface RCN 0 (Def) sort D3.4 Calculate bin sizes RCO 0 (Def) all System interface controlling phase 0 (Def) (and linkage editor) RCP 0 (Def) all D3.4 List exit routines to be linkage edited RCQ 0 (Def) all Specify input area for control statements RCR 0 (Def) sort D3.4 Calculate B and G for tape sort and choose RCS D3.4 Optimal technique; interpret MODS statement RCC 1,2,3 all Free storage between phases RCU 0 (Def+Opt) all Contain error messages for phase 0 RCV 1,2,3 all System interface for processing records RCX 0 (Def) all Contain messages for MODS card errors RCZ 0 (Opt) all D3.4 Move CPI information to PPI RC2 0 (Def) all Calculation for extract		0 (Def) all	D3.4	Scan MODS control statement
and their characteristics RCJ 0 (Opt) sort Check disk capacity RCK 0 (Opt) sort D3.4 BALN (disk): optimize B and G RCL 0 (Def) merge D3.4 Merge-only optimization RCM 0 (Def) all D3.4 System interface RCN 0 (Def) sort D3.4 Calculate bin sizes RCO 0 (Def) all System interface controlling phase 0 (Def) (and linkage editor) RCP 0 (Def) all D3.4 List exit routines to be linkage edited RCQ 0 (Def) all Specify input area for control statements RCR 0 (Def) sort D3.4 Calculate B and G for tape sort and choose RCS D3.4 Optimal technique; interpret MODS statement RCT 1,2,3 all Free storage between phases RCU 0 (Def+Opt) all Contain error messages for phase 0 RCV 1,2,3 all System interface for processing records RCX 0 (Def) all Contain messages for MODS card errors RCZ 0 (Opt) all D3.4 Move CPI information to PPI RCC 0 (Def) all Calculation for extract		0 (Dof) 311	20.4	Thentifu input sutput and work data cote
RCK 0 (Opt) sort D3.4 BALN (disk): optimize B and G RCL 0 (Def) merge D3.4 Merge-only optimization RCM 0 (Def) all D3.4 System interface RCN 0 (Def) sort D3.4 Calculate bin sizes RCO 0 (Def) all System interface controlling phase 0 (Def) (and linkage editor) RCP 0 (Def) all D3.4 List exit routines to be linkage edited RCQ 0 (Def) all Specify input area for control statements RCR 0 (Def) sort D3.4 Calculate B and G for tape sort and choose RCS D3.4 Optimal technique; interpret MODS statement RCT 1,2,3 all Contain error messages for phase 0 RCU 0 (Def+Opt) all System interface for processing records RCX 0 (Def) all System interface for processing records RCX 0 (Def) all Contain messages for MODS card errors RCX 0 (Opt) all D3.4 Move CPI information to PPI RC2 0 (Def) all Calculation for extract	KC1	o (Der) arr		
RCL 0 (Def) merge D3.4 Merge-only optimization RCM 0 (Def) all D3.4 System interface RCN 0 (Def) sort D3.4 Calculate bin sizes RCO 0 (Def) all System interface controlling phase 0 (Def) (and linkage editor) RCP 0 (Def) all D3.4 List exit routines to be linkage edited RCQ 0 (Def) all Specify input area for control statements RCR 0 (Def) sort D3.4 Calculate B and G for tape sort and choose RCS RCS D3.4 Optimal technique; interpret MODS statement RCT 1,2,3 all Free storage between phases RCU 0 (Def+Opt) all Contain error messages for phase 0 RCV 1,2,3 all System interface for processing records RCX 0 (Def) all Contain messages for MODS card errors RCX 0 (Opt) all D3.4 Move CPI information to PPI RC2 0 (Def) all Calculation for extract				Check disk capacity
RCM 0 (Def) all D3.4 Calculate bin sizes RCO 0 (Def) all System interface controlling phase 0 (Def)				BALN (disk): optimize B and G
RCM 0 (Def) all D3.4 Calculate bin sizes RCO 0 (Def) all System interface controlling phase 0 (Def) (and linkage editor) RCP 0 (Def) all D3.4 List exit routines to be linkage edited Specify input area for control statements O (Def) sort D3.4 Calculate B and G for tape sort and choose RCS D3.4 Optimal technique; interpret MODS statements CO (Def+Opt) all Contain error messages for phase 0 (Def+Opt) all Contain error messages for phase 0 (Def) all Contain messages for MODS card errors CO (Opt) all D3.4 Move CPI information to PPI Calculation for extract	RCL	0 (Def) merg	e D3.4	Merge-only optimization
RCO 0 (Def) sort D3.4 Calculate bin sizes RCO 0 (Def) all System interface controlling phase 0 (Def) (and linkage editor) RCP 0 (Def) all D3.4 List exit routines to be linkage edited RCQ 0 (Def) all Specify input area for control statements RCR 0 (Def) sort D3.4 Calculate B and G for tape sort and choose RCS D3.4 optimal technique; interpret MODS statement RCT 1,2,3 all Free storage between phases RCU 0 (Def+Opt) all Contain error messages for phase 0 RCV 1,2,3 all System interface for processing records RCX 0 (Def) all Contain messages for MODS card errors RCX 0 (Opt) all D3.4 Move CPI information to PPI RC1 0 (Opt) all D3.4 Move CPI information to PPI RC2 0 (Def) all Calculation for extract	RCM			System interface
(and linkage editor) RCP 0 (Def) all D3.4 List exit routines to be linkage edited RCQ 0 (Def) all Specify input area for control statements RCR 0 (Def) sort D3.4 Calculate B and G for tape sort and choose RCS D3.4 optimal technique; interpret MODS statement RCT 1,2,3 all Free storage between phases RCU 0 (Def+Opt) all Contain error messages for phase 0 RCV 1,2,3 all System interface for processing records RCX 0 (Def) all Contain messages for MODS card errors RCX 0 (Opt) all Control of phase 0 (Opt) RC1 0 (Opt) all D3.4 Move CPI information to PPI RC2 0 (Def) all Calculation for extract				
RCP 0 (Def) all D3.4 List exit routines to be linkage edited Specify input area for control statements O (Def) sort D3.4 Calculate B and G for tape sort and choose O (Def) sort D3.4 Optimal technique; interpret MODS statements of the storage between phases O (Def+Opt) all Contain error messages for phase O (Def) all Contain messages for modes of the storage of the	RCO	0 (Def) all		
RCQ 0 (Def) all Specify input area for control statements Calculate B and G for tape sort and choose Calculate	RCP	0 (Def) all	D3-4	
RCR 0 (Def) sort D3.4 Calculate B and G for tape sort and choose RCS D3.4 optimal technique; interpret MODS statemer RCT 1,2,3 all Free storage between phases RCU 0 (Def+Opt) all Contain error messages for phase 0 RCV 1,2,3 all System interface for processing records RCX 0 (Def) all Contain messages for MODS card errors RCZ 0 (Opt) all D3.4 Move CPI information to PPI RC1 0 (Def) all Calculation for extract			,	Specify input area for control statements
RCT 1,2,3 all Free storage between phases RCU 0 (Def+Opt) all Contain error messages for phase 0 RCV 1,2,3 all System interface for processing records RCX 0 (Def) all Contain messages for MODS card errors RCZ 0 (Opt) all Control of phase 0 (Opt) RC1 0 (Opt) all D3.4 Move CPI information to PPI RC2 0 (Def) all Calculation for extract			D3.4	
RCU 0 (Def+Opt) all Contain error messages for phase 0 RCV 1,2,3 all System interface for processing records RCX 0 (Def) all Contain messages for MODS card errors RCZ 0 (Opt) all Control of phase 0 (Opt) RC1 0 (Opt) all D3.4 Move CPI information to PPI RC2 0 (Def) all Calculation for extract			D3.4	
RCV 1,2,3 all System interface for processing records RCX 0 (Def) all Contain messages for MODS card errors RCZ 0 (Opt) all Control of phase 0 (Opt) RC1 0 (Opt) all D3.4 Move CPI information to PPI RC2 0 (Def) all Calculation for extract				
RCX 0 (Def) all Contain messages for MODS card errors RCZ 0 (Opt) all Control of phase 0 (Opt) RC1 0 (Opt) all D3.4 Move CPI information to PPI RC2 0 (Def) all Calculation for extract	RCU	0 (Def+Opt) a	11	
RCX 0 (Def) all Contain messages for MODS card errors RCZ 0 (Opt) all Control of phase 0 (Opt) RC1 0 (Opt) all D3.4 Move CPI information to PPI RC2 0 (Def) all Calculation for extract	RCV			System interface for processing records
RCZ 0 (Opt) all Control of phase 0 (Opt) RC1 0 (Opt) all D3.4 Move CPI information to PPI RC2 0 (Def) all Calculation for extract				
RC2 0 (Def) all Calculation for extract				
RC2 0 (Def) all Calculation for extract	RC1	0 (Opt) all	D3.4	Move CPI information to PPI
			=	
	RC3	0 (Def) all		Contains messages for MODS card scanning

I

CSECT	Used in	MO	Purpose
Name	Phase:	Diag	102600
RC4	0 (Opt) sort		Calculate and optimize disk work areas
	·		
RC6	1	N3.1	Determine modules needed in phase 1
RC7	2 N3	.2,N3.1	Determine modules needed in phase 2
RC8	3 all	N3.3	Determine modules needed in phase 3
RC9	1,2,3 all		Load necessary modules for each phase
		N3.2,	
		N3.1	
		2.20.	
RDB	1	N3.1	Deblock fixed-length records with in-line
-1.2-2	•		move (no exits)
RDC	1	N3.1	Deblock fixed-length records with link to
	•		multiple move (no exits)
RDD	1	N3.1	Deblock fixed-length records (exits used)
RDE	i	N3.1	Deblock var. records (exits used)
RDG	i	N3.1	Deblock var. records (no exits)
KDG	•	N3.1	Deprock Agr. records (no extra)
RDH	3 sort	из.3	Deblock fixed-length records (for disk and
KDH	3 SOIL	M2.2	read-backward tape)
DDT	3	N2 2	Tead-backward tape)
RDI	3 sort	из.3	Deblock var. records
RDJ	3 merge	N3.3	Deblock
RDL	2		Set up deblock area
			Onon a com 1-11-1-1 (1 -1 1-1 (1 -1 1-1)
RDP	1	N3.1	CRCX & OSCL: deblock fixed-length records up
550			to 256 bytes (no exits)
RDQ	1	N3.1	CRCX & OSCL: deblock fixed-length records
	4		up to 256 bytes (no exits)
RDR	1	N3.1	CRCX & OSCL: deblock fixed-length records
		_	(exits used)
RDS	1	N3.1	CRCX & OSCL: deblock variable records
			(exits used)
RDT	1	N3.1	CRCX & OSCL: deblock var. records (no exits)
	•		
RDU	3 merge	N3.3	Deblock (exits used)
RDX	3 sort	N3.3	Deblock fixed-length records
			(for read-forward tape)
	_		
RGB	2	N3.2	Read tape backwards
RGC	2	N3.2	Read disk
RGD	3 sort	из.3	Read tape backwards
RGE	3 sort	N3.3	Read disk
RGF	3 merge		Indicate input end-of-file
RGL	2	N3.2	Read tape forward
RGM	3 sort	N3.3	Read tape forward
-	-	- · · ·	•
RGV	1 sort,	N3.1,	VSAM read
	3 merge	N3.3	
RMA	1		Contain messages for ph1 running modules
RMB	2		Contain messages for ph2 running modules
RMC	3 all		Contain messages for ph3 running modules
	-		grant and g
ROA	1	N4.11	Sort fixed-length records, 'multiple'
			(not POLY)
ROB	1	N4.11	Sort fixed-length records, 'single'
		J	(not POLY)
ROC	1	N4.11	
ROD	i		Sort var. records, 'single' (not POLY)
ROE	1		POLY: sort fixed-length records, 'multiple'
ROF	i		POLY: sort fixed-length records, 'single'
ROG	i	N4.11	POLY: sort var. records, 'multiple'
		- - -	

	Used in Phase:	MO Diag	Purpose
ROH	1	N4.11	POLY: sort var. records, 'single'
ROI	1	N4 - 12	BALN (tape): sort algorithm
ROJ	i	N4.12	POLY: sort algorithm
ROK	i	N/L 12	POLY: sort algorithm BALN (disk): sort algorithm
RON	i	N4. 12	OSCL: sort algorithm; initiate checkpoint
KON	•	N4.12,	
ROP	2 & 3 all	N3.2, N3.3	Merge records 'multiple'
ROQ	2 & 3 all		Merge records, 'single'
ROR	2	N3.2	BALN (tape): merge algorithm
ROS	2	N3.2	POLY: merge algorithm
ROT	2	из 5	POLY: merge algorithm BALN (disk): merge algorithm
		M3.2	
ROW	1		CRCX & OSCL: sort fixed-length records, 'multiple'
ROX	1		CRCX & OSCL: sort fixed-length records, 'single'
ROY	1		CRCX & OSCL: sort var. records, 'multiple'
ROZ	i		CRCX & OSCL: sort var. records, 'single'
RPA	1	N4.12, N3.2	Write tape
RPB	1		Write disk
RPC	1	N3.1	
RPD	2	N3.2	Write tape
RPE	2		Write disk
	2	N3.2	End-of-phase housekeeping (not CRCX
RPF	2	M2.Z	or OSCL)
RPG		из.3	Open output; end-of-phase housekeeping
RPM	1 & 2	N3 2	procedures OSCL: end-of-phase housekeeping
	3 all	N3.3	VSAM write
RPV	2 gil	N3.3	VSAM WIIte
8BN	3 all	N3.3	Block var. spanned records (no exits)
8BO	3 all	N3.3	Block var. spanned records (exits used)
8CI	0 (Def) all	•	Contain tables and constants for RCI
8CK	0 (Opt) sort		BALN (disk) timing estimate routine
8CR	3 merge		Checkpoint routine at end of output volume
8DJ	3 merge	N3.3	Deblock variable-length spanned records (no exits)
		_	
8GB	2	N3.2	
8GC	3 sort	из.3	
80N	1		CRCX sort algorithm
8PA	1 & 2		CRCX write
8PM	1 & 2	N3.2	CRCX end-of-phase housekeeping
9BN	3 all	17.6	Block var. spanned records, no exits
			(assignment)
9 B O	3 all.		Block var. spanned records, exits used (assignment)

CSECT Name	Used in Phase:	MO D ia g	Purpose
9DJ	3 merge		Assignment deblock variable-length spanned records (no exits)
9GB 9GC 9GN	2 3 sort 1	N3.1	CRCX read assignment CRCX read assignment CRCX: generate DCBs, OBs, ECBs, alternate CCW pointers
90N 9PA	1 1 & 2		CRCX algorithm assignment CRCX write assignment

Section 5: Data Areas

The program makes extensive use of a large data area. For Blockset applications the area is called COMMON, and its layout is defined in the DSECT COMMON, for Peerage and Vale applications the area is called COMMA, and its layout is defined in the DSECT ICECOMMA. For other applications the information in COMMA is moved in phase 0 to a 'control phase information area', the CPI, and thence to a 'phase-to-phase information area', the PPI, the DSECT for which resides in module ICERCA. How to list COMMA is described in Section 6. The CPI and PPI are described in detail below together with Module ICEAM1, which contains CSECT ICEAM1, which contains the default values specified when the program was installed.

For the CPI and PPI a listing is given showing the position, length, name and contents of each field. For the PPI a 'map' is also provided, as well as an index.

If you know the name of a field in the CPI or the PPI, but not its position, you can find its displacement from the cross-reference tables in Section 6.

| Blockset Area (COMMON)

| This area is allocated by ICEMAN and is used by all Blockset modules. | It is described in the DSECT COMMON. The program uses register 13 as a base register for COMMON.

COMMON PRINTOUT

```
DISPL NAME
                                                                           DESCRIPTION
          0000 COMMON DSECT
                                       SAVE AREA
                                                                             START OF OS SAVE AREA
LEVEL ZERO SAVE AREA
LEVEL ONE SAVE AREA
LEVEL TWO SAVE AREA
LEVEL THREE SAVE AREA
                        DS
                                       3F
0000
000C COMSAVEO DS
                                       15F
0048 COMSAVE1 DS
007C COMSAVE2 DS
00B0 COMSAVE3 DS
                                       13F
                                       13F
                                       13F
                                       FOUNDATION
00E4 COMFOUND DS
                       PRESET DEFAULTS
00E4 COMFAULT DS
                        VERSION NUMBER OF COMMON
COMVERSE EQU
00E4 COMVERZN DS
                                                                                CURRENT COMMON VERSION NUMBER
                                                                                CONTAINS COMVERSE
                       MODE INDICATORS
00E6 COMVOKED DS
00E7 COMMODE DS
00E8 COMCOBIN DS
00E9 COMCOBUT DS
                                                                               USER INVOKED STATUS
SORT OR MERGE MODE INDICATOR
COBOL INPUT MCDE OPTION
COBOL OUTPUT MODE OPTION
SORT RECORD FORM
                                       C
                                        C
                                        C
                                       C
00EA COMRECEM DS
00EB COMMNOGO DS
00EC COMABEND DS
                                                                               ERROR ENCOUNTERED INDICATOR ABDUMP DD ENCOUNTERED
                                       C
                                                                              ABDUMP DD ENCOUNTERED
SYSTEM TYPE FROM CVT
TEST STATUS INDICATOR
ISSUE 190 MESSAGES
SEQUENCE CHECK SORTOUT
SNAP WRITEBACK BLOCKS
VERIFY BIN COUNT
TRACE INPUT BLOCKS
ISSUE PHASEO MSGS
COMMON IN/OUT DATA SET
V = VIRTUAL, R = REAL
END OF VOLUME ON SORTIN INDICATOR
END OF FILE ON SORTIN INDICATOR
SECONDARY KEYS PRODUCED
'L' = LOW, 'H' = HIGH.
'Y' = SKIP INTERMEDIATE MERGE
OOED COMSYSTM DS
                                       Č
OOEE COMMTEST DS
                                        č
          COMMT190 EQU
COMMTSEQ EQU
                                       128
                                        64
          COMMSNAP EQU
                                        32
          COMMTVER EQU
                                        32
                           EQU
          COMFLAG
                                       16
COMMITPHO EQU
COMMITPHO EQU
COMEDITYP DS
COMEDITYP DS
COMEDITYP DS
                                       800000
00F2 COMEOF DS
00F3 COMSKEYS DS
00F4 COMPMODE DS
00F5 COMSKIPM DS
                       MESSAGE CSECT
00F8 COMDIAGS DS
                                                                               DIAGNOSTIC LIST ADDRESS
                       CONSTANTS
OOFC COMINUSA DS
0100 COMINUSA DS
                                                                               MINUS 4
MINUS 8
```

5PL	. NAME		DESCRIPTION
	x	USER SPECIFIED OPTI	ONS
14	COMOPTNS DS	05	
. .	× ALTERNA	TE SORT NAME	
04	COMMXCTL DS	CL8	NAME TO XCTL TO
O C	COMOPWTO DS	CL12	WTO HEADER AND ROUTING
18	COMAXLIM DS	F	MAX FOR SIZE=MAX
1 C 2 O	COMMAIN DS COMSERVE DS	F	NAME TO XCTL TO WTO HEADER AND ROUTING MAX FOR SIZE=MAX MAX FOR SIZE= NOT SPECIFIED AMOUNT OF RESERVED MEMORY 'SORT' DD PREFIX OR ALIAS SYSOUT FILE NAME COBOL SYSOUT=A DDNAME
	* DDNAME	OPTIONS	
24 28	COMSPELX DS	CL4 CLR	'SORT' DD PREFIX OR ALIAS
30	COMCSYSO DS	CL8	COBOL SYSOUT=A DDNAME
	* SORTIN/S	SORTOUT MEMBER NAMES	
	COMEMBRO EQU	COMCSYSO	SORTIN MEMBER SORTOUT MEMBER PRINT MESSAGE CLASS COBOL SORT MESSAGE PRINT CLASS TYPE MESSAGE CLASS LOG MESSAGE CLASS INITIAL LINE COUNTER VALUE LINES PER PAGE LIST CONTROL CARDS OPTION
38	COMPRINT DS	C CLH33 ULITUM3	PRINT MESSAGE CLASS
39	COMCPRNT DS	Č	COBOL SORT MESSAGE PRINT CLASS
S A	COMTYPE DS	C	TYPE MESSAGE CLASS
7 13	X LISTING	CONTROL COUNTS	LUG MESSAGE CLASS
3 C	COMLCTR DS	C	INITIAL LINE COUNTER VALUE
3 D	COMLINES DS	CARD LIET OPTION	LINES PER PAGE
3 E	COMSGLST DS	CAKD LIST OFILON	LIST CONTROL CARDS OPTION
	V MEAGLAE	TOUT OFFICE	
3F	COMPFIX DS	CL3	PREFIX NAME
12 43	COMJOB DS	C	CONTROL CARD ERROR POSTION FLAG TYPE/LOG JOB NAME
44	COMSTEP DS	CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC	TYPE/LOG STEP NAME
	X ADRUKTAL	LERCHNALING HELLIN	¬
4 <i>5</i>	COMIDUMP DS	C	DUMP OPTION FOR IREND
47	COMOBEND DS	Č	ABEND IF ERROR DURING INITIAL PHASE DUMP OPTION FOR IBEND ABEND IF ERROR DURING OTHER PHASE DUMP OPTION FOR OBEND
48	COMODUMP DS	C CK AREA MODE INDICAT	DUMP OPTION FOR OBEND
49	COMLPA DS	C AREA MODE INDICAT	UK TPUT AND OPUT ARE IN 19A
	* CONSORT	SVC	THE DISTRICT OF THE PARTY OF TH
4 A	COMSVC DS	H	SVC FOR SORT
	X		212C OF OFITOM2
	× ×	DCB POINTERS	OR IPUT AND OPUT ARE IN LPA SVC FOR SORT SIZE OF OPTIONS ADDRESS OF SYSOUT DCB ADDRESS OF SYSIN DCB
4 C	COMPRDCB DS	F F	ADDRESS OF SYSOUT DCB ADDRESS OF SYSIN DCB
50 54	COMRDDCB DS	F	ADDRESS OF SYSIN DCB SORTIN DCB ADDRESS
58	COMUTDOB DS	F	SORTOUT DCB ADDRESS
5C	COMWKDCB DS	F	SORTWK OPEN LIST ADDRESS
50	COMOIDCE DS * UNIT RECO	F D DEADE	SORTING1 OPEN LIST ADDRESS
54	COMPRDEC DS	F DECBS	ADDRESS OF SYSOUT DEC
68	COMRDDEC DS	F	SYSIN DECB ADDRESS
	×	ADDRESSES OF VARIOUS	S GENERATED ROUTINES IN COMMON
5 C	COMLOAD DS COMUNLD DS COMKLOAD DS	F	LOAD SUBROUTINE ADDRESS
ZÓ	COMUNLD DS	<u>F</u>	UNLOAD SUBROUTINE ADDRESS
/4 72	COMKLOAD DS COMKUNLD DS	F F	KEY LOAD SUBROUTINE ADDRESS KEY UNLOAD SUBROUTINE ADDRESS
7 C	COMOVREC DS COMLDKAD DS	F	ADDRESS OF RECORD MOVE ROUTINE
ΒÓ	COMLDKAD DS	Ē	ADDRESS OF KAD LOAD ROUTINE
54	COMSTORE DS	F	RESTORE CODE ADDRESS

DISPL	NAME			DESCRIPTION
				FIELDS *
018C 018E 0190 0192 0194	COMIFORM COMOFORM COMIBLKZ COMOBLKZ COMIRECZ COMORECZ COMIBLKL COMVOLCT * OT COMINIO	DS DS DS DS DS DS HER SO	C C H H	INPUT RECFM OUTPUT RECFM INPUT BLOCKSIZE OUTPUT BLOCKSIZE INPUT LRECL OUTPUT LRECL INPUT FILE AVERAGE BLOCK LENGTH SORTIN VOLUME COUNT MINIMUM SORTIO BUF TRANSFER COUNT MAXIMUM SORTIO BUF TRANSFER COUNT
	×	-	RECORD COUNTS	**************************************
019C 01A0 01A4 01A8 01AC 01B0 01B4 01B8	COMFILEZ COMIRECS COMORECS COMARECS COMDRECS COMRUNSZ COMVALCT COMVALPR	DS DS DS DS DS DS	F F F F F F	FILE SIZE INPUT REC COUNT OUTPUT REC COUNT ADDED REC COUNT DELETED REC COUNT CURRENT RUN LENGTH BVAL STANDARD COUNT BVAL PRIOR COUNT
	×		OTHER COUNTS	
018C 01C0 01C0 01C4 01C8 01C8	COMMAINO COMFREEB COMMAIN1 COMBINSN COMRUNS COMMAIN2 COMMAIN2 COMIRSAZ COMIRSAZ	DS DS DS DS DS DS	F	SAVED INIT MAIN STORAGE VALUE FREE BIN COUNT SAVED MAIN STORAGE VALUE SELECTED BIN RECORD COUNT RUNS DISTRIBUTED BY INPUT PHASE USED MAIN STORAGE VALUE RSA SIZE INPUT RSA SIZE PRIMARY KAD COUNT
	×		MEMORY MANAGEMENT F	:
01D4 01D8	COMFILED !	DS DS	EA POINTERS F F DRAGE FIELDS	GET FILE AREA PTR PUT FILE AREA PTR
01E0 01E4 01E8 01EC 01F0 01F4	COMMAREA COMMEND COMAVAIL COMSAD COMSPACE COMINGET COMINCOR COMINCOR COMSHORT	DS DS DS DS DS DS DS	F F F F F F	ADDRESS OF COMMON WORK AREA ADDRESS OF END OF COMMON WORK AREA ADDRESS OF AVAILABLE COMMON ADDRESS OF FIRST STORAGE AREA DESC BYTES AVAILABLE MINIMUM GETMAIN AMOUNT MINIMUM SPACE NEEDED TO SORT MAIN STORAGE DEFICIT
	× ×		COMPARE STRUCTURE DA	X X
0200 0204	COMLISTA DE COMLISTA DE COMTREE DE COMTSIZE DE COMTSIZ	DS DS	F F F	LIST HEADERS FOR MERGE SORT. LOSER TREE AREA OFFSET TO LAST NODE

ISPL	NAME		DESCRIPTION
	XXXXXXX		
20C 210 214	COMBLKSI DS COMRPB DS COMKPB DS	F F F	WORKFILE BLOCK SIZE RECORDS PER WORKFILE BLOCK KEYS PER WORKFILE BLOCK SIZE OF BIN KAD PREFIX SIZE OF KAD PREFIX SIZE OF KAD BLOCK ENTRY
218 21¢ 220	COMKPFIX DS COMKDATA DS COMKBLKZ DS	F F F	SIZE OF BIN KAD PREFIX SIZE OF KAD PREFIX SIZE OF KAD BLOCK ENTRY
	*	I/O TRANSFER AREA	A FIELDS
224 228 220 230 234	X	F F F F	ADDRESS OF IOX LENGTH OF IOX LOW I/O ADDRESS SORTIO DATA ADDRESS IN IOX SORTWK DATA ADDRESS IN IOX
	×	DXL LISTS	
238 23C 240 244 248	COMDXLSQ DS COMDXLIO DS COMDXLWQ DS COMDXLRQ DS COMDXKEY DS	F F F F	SEQUENTIAL DXL LIST HEAD SORTIO CIRCULAR LIST ENTRY SORTWK DXL WRITE QUEUE SORTWK DXL READ QUEUE ADDRESS OF KEY AREA DXL
	*	DITEILES	
1250 1258 1258 12560 12668 12278 12278 12284 12288 12288 12299	COMAREAQ DS COMBINFQ DS COMBINFQ DS COMLADRQ DS COMLADRQ DS COMBINST DS COMBINST DS COMKEYFQ DS COMCADAQ DS COMCADAQ DS COMBVDFQ DS COMBVDFQ DS COMBVDFQ DS COMBVDFQ DS COMKEYPT DS COMKEYPT DS COMKEYPQ DS COMKEYPQ DS COMKIRRF DS COMKIRRQ DS		WORK AREA PREFERENCE QUEUE QUEUE OF AVAILABLE RECORD BINS LAD AVAILABLE LIST NEXT STRING LAD QUEUE QUEUE OF EXCP IOBS SELECTED BIN QUEUE SELECTED QUEUE TAIL FREE KEY HOLDER QUEUE FREE CAD QUEUE AVAILABLE CYLINDER AREA QUEUE FREE BVD QUEUE ALLOCATED BVD QUEUE ADDRESS OF LOW BOUNDARY VALUE DEF ADDRESS OF HIGH BOUNDARY VALUE DEF PRIMARY KEY QUEUE PRIMARY KEY QUEUE KAD IRRR HOLDER FREE QUEUE
1298	*	F KEY MISCELLANEOU:	KAD IRRR HOLDER QUEUE
29C 2A0 2A4 2A8	COMOVALT DS COMTRACT DS COMEQUAL DS COMEXITS DS	F	
	×	VARIOUS FULLWORD	SIZES
02B0 02B4 02B8	COMBINSZ DS COMMRECZ DS COMDXLSZ DS COMPAREZ DS COMIORPB DS	F	SIZE OF A RECORD BIN SIZE OF A WORKFILE RECORD SIZE OF DXL (DEPENDS ON V OR R) LENGTH OF COMPARE SORTIO REC PER BLOCK

DISPL	. NAME		DESCRIPTION	_
	X	TIOT ADDRESSES		×
02C0 02C4 02C8	COMTIOT DS COMSAVEI DS COMSAVEO DS	F F RITO FIFIDS	ADDRESS OF TIOT TIOT ENTRY FORT SORTIN TIOT ENTRY FOR SORTOUT	×
02CC 02D0 02D2 02D4 02D6	COMCCHH DS COMRECNO DS COMTRBAL DS COMTRSAV DS COMXTENT DS	F 2C H H	ADDRESS OF TIOT TIOT ENTRY FORT SORTIN TIOT ENTRY FOR SORTOUT LATEST SORTIO CCHH LATEST R CURRENT TRACK BALANCE PREVIOUS TRACK BALANCE CURRENT EXTENT	
	XX	WTO AREA		×
			HEADER FOR L FORM OF WTO TEXT OF WTO MESSAGE PREFIX MESSAGE NUMBER	
0354	COMMMTAG EQU COMWJOB EQU COMWSTEP EQU COMWMESS EQU COMWROUT DS	COMWMNUM+3 COMWMTAG+2 COMWJOB+9 COMWSTEP+9 F	MESSAGE TAG JOB NAME STEP NAME MESSAGE ROUTING FOR WTO	
	X	MISCELLANEOUS		××××
0358 0360 0364 0368 036C	COMDUBBL DS COMOLIST DS COMOLEND DS COMSQRT DS COMIDKEY DS	D F F F	OUBLEWORD ALIGNED WORK AREA OPEN LIST ADDRESS END OF OPEN LIST SQUARE ROOT OF RPB MIDDLE KEY	
	COMISCEL DS COMISREL EQU COMXSTAT EQU COMXE15 EQU COMXE35 EQU COMXE01 EQU	ANEOUS BIT SWITCHES 4C 1 COMISCEL+1 1 2 4 16 SCELLANEOUS	SWITCH VALUES 0 => RELEASE SORTWORK SPACE EXIT STATUS E15 IS PRESENT E35 IS PRESENT E01 IS PRESENT E03 IS PRESENT	
0374 0378 037C 0380 0384 0388 038C	COMBVD DS COMLADS DS COMLADLM DS COMIIOB DS COMOIOB DS COMSIRR DS COMSIRR DS COMBOUND DS COMLIMIT DS	F F F	CURRENT BVD LAD AREA ADDRESS LAD AREA LIMIT INPUT IOB OUTPUT IOB ABNORMAL CHANNEL END SECONDARY KEYFILE IRRR COUNT OF WITHIN BOUND RECORDS BOUND LIMIT	
	×	KEYETLE COUNTS AND	LIMITS	¥
039A 039C 039E 03A0 03A2	COMKEYPC DS COMKEYSC DS COMKEYPN DS COMKEYPX DS COMKEYPX DS COMINKPB DS COMAXKPB DS	H H H H H	PRIMARY KEY COUNT SECONDARY KEY COUNT PRIMARY KEY MINIMUM PRIMARY KEY MAXIMUM SECONDARY KEY MAXIMUM MINIMUM KEYS PER BLOCK MAXIMUM KEYS PER BLOCK	~

MISCELLANEOUS HALFWORD FIELDS	DISPL	. NAME		DESCRIPTION		
SAAB COMMERCZ DS		* MISCELLANEOUS HALFWORD FIFEDS				
X		×	UT2CELCAVEOO2 UMPL	WORD FIELDS		
X	03A6 03A8 03AA 03AC 03AE 03B0 03B2	COMKRECZ DS COMINBOF DS COMKDELT DS COMKDELT DS COMMUNALS DS COMMUNALS DS COMMUNALS DS	H H H H K H	KEY RECORD LENGTH BUFFER OFFSET FOR ASCII SORTIN PRIMARY KEY MULTIPLE LOAD DELTA 1ST TWO BYTES OF JFCLSTAR WORKFILE DXLS LOGICAL CYLINDER ASSIGNMENT CTR PRIOR BLOCK SIZE		
X	03B4	COMBULIM DS	n H	HIGH BV XFER COUNT		
0.388 COMPMOVE DS		¥	KEY MOVES AND COMPA	AKPS ~ SPP LAKIP IN KPYMYLLL		
03E8 COMPARE DS CL6 CLC 0(,R14),0(R15) 03E8 COMPSLAB DS CL6 CLC 4(,R4),4(R5) COMPSLAK EQU COMPCOPA COMPSLAK EQU COMPCOPB 03F4 COMPALTA DS CL6 CLC 4(,R2),0(R4) 03F4 COMPALTB DS CL6 CLC 4(,R3),0(R4) 0400 COMOVKAD DS CL18 MVC 0(KADLIST,R1),0(R15) ** BLOCK READ CHECK FIELDS ** BLOCK READ CHECK FIELDS ** BLOCK READ CHECK FIELDS ** BLOCK COUNTER 0418 COMBLOKO DS F OUTPUT BLOCK COUNTER 0410 COMBLOKI DS F INPUT BLOCK COUNTER 0410 COMBLOKI DS F SUM OF OUTPUT COUNT IDS 0424 COMPANDY DS F SECONDARY ALLOCATION WORD 0428 COMMAX DS F SECONDARY ALLOCATION WORD 0428 COMMON DS F SECONDARY ALLOCATION WORD 0428 COMMON DS F SECONDARY ALLOCATION WORD 0428 COMCLOSE DS F CLOSE LIST AREA COMIDSHM EQU COMCLOSE INPUT DATA SET NAME ** PHASE LOCAL WORK AREA 042C COMLOCAL DS 22F ** ** ** ** ** ** ** ** **	03B8 03B8	COMPMOVE DS COMXCRUN DS	0H CL6	XC 0(,R1),0(R1)		
03E8 COMPARE DS CL6 CLC 0(,R14),0(R15) 03E8 COMPSLAB DS CL6 CLC 4(,R4),4(R5) COMPSLAK EQU COMPCOPA COMPSLAK EQU COMPCOPB 03F4 COMPALTA DS CL6 CLC 4(,R2),0(R4) 03F4 COMPALTB DS CL6 CLC 4(,R3),0(R4) 0400 COMOVKAD DS CL18 MVC 0(KADLIST,R1),0(R15) ** BLOCK READ CHECK FIELDS ** BLOCK READ CHECK FIELDS 0418 DS 0D 0418 COMBLOKO DS F OUTPUT BLOCK COUNTER 041C COMBLOKI DS F INPUT BLOCK COUNTER 041C COMBLOKI DS F SUM OF OUTPUT COUNT IDS 0424 COMMAX DS F SUM OF OUTPUT COUNT IDS 0428 COMMAX DS F SECONDARY ALLOCATION WORD 0428 COMMAX DS F SECONDARY ALLOCATION WORD 0428 COMMAX DS F SECONDARY ALLOCATION WORD 0428 COMMONA DS F SECONDARY ALLOCATION WORD 0428 COMMONA DS F SECONDARY ALLOCATION WORD 0428 COMMONA DS F CLOSE LIST AREA COMIDSHM EQU COMCLOSE INPUT DATA SET NAME ** PHASE LOCAL WORK AREA 042C COMLOCAL DS 22F ** MERGE/SORT INITIALIZATION ROUTINE 0484 COMERGED DS 28F COMIDSHM EQU COMCLOSE INPUT DATA SET NAME ** TREE/MERGE ROUTINE SPACE ** MERGE/SORT ROUTINE 0484 COMERGES DS 48F COMTABLE EQU COMERGES -4 TRANSLATE/SIZE TABLE ** TREE INITIALIZATION ROUTINE 0584 COMTREED DS 36F ** TREE ADD ROUTINE 0584 COMTREED DS 36F ** TREE ADD ROUTINE 0684 COMTREED DS 16F	03C4	COMPAB DS	CL6	CLC 8(,R4),0(R3)		
03E8 COMPARE DS CL6 CLC 0(,R14),0(R15) 03E8 COMPSLAB DS CL6 CLC 4(,R4),4(R5) COMPSLAK EQU COMPCOPA COMPSLAK EQU COMPCOPB 03F4 COMPALTA DS CL6 CLC 4(,R2),0(R4) 03F4 COMPALTA DS CL6 CLC 4(,R3),0(R4) 0400 COMOVKAD DS CL18 MVC 0(KADLIST,R1),0(R15) ** BLOCK READ CHECK FIELDS ** BLOCK READ CHECK FIELDS ** BLOCK READ CHECK FIELDS ** BLOCK COMPOVED S F OUTPUT BLOCK COUNTER 0418 COMBLOKO DS F INPUT BLOCK COUNTER 0410 COMBLOKI DS F SUM OF OUTPUT COUNT IDS 0420 COMMAND DS F SECONDARY ALLOCATION WORD 0420 COMMAND DS F SECONDARY ALLOCATION WORD 0424 COMMOND DS F SECONDARY ALLOCATION WORD 0428 COMMOND DS F CLOSE LIST AREA COMIDSNM EQU COMCLOSE INPUT DATA SET NAME ** PHASE LOCAL WORK AREA 042C COMLOCAL DS 22F ** MERGE/SORT INITIALIZATION ROUTINE 0484 COMERGEI DS 28F ** MERGE/SORT ROUTINE 0484 COMERGES DS 48F COMTABLE EQU COMERGES -4 TRANSLATE/SIZE TABLE ** TREE INITIALIZATION ROUTINE 0584 COMTREED DS 36F ** TREE ADD ROUTINE 0584 COMTREED DS 36F ** TREE ADD ROUTINE 0644 COMTREED DS 36F ** TREE DELETE ROUTINE 0646 COMTREED DS 16F	03CA	COMPCOPA DS	CL6	CLC 4(,R4),0(R14)		
03E8 COMPARE DS CL6 CLC 0(,R14),0(R15) 03E8 COMPSLAB DS CL6 CLC 4(,R4),4(R5) COMPSLAK EQU COMPCOPA COMPSLAK EQU COMPCOPB 03F4 COMPALTA DS CL6 CLC 4(,R2),0(R4) 03F4 COMPALTA DS CL6 CLC 4(,R3),0(R4) 0400 COMOVKAD DS CL18 MVC 0(KADLIST,R1),0(R15) ** BLOCK READ CHECK FIELDS ** BLOCK READ CHECK FIELDS ** BLOCK READ CHECK FIELDS ** BLOCK COMPOVED S F OUTPUT BLOCK COUNTER 0418 COMBLOKO DS F INPUT BLOCK COUNTER 0410 COMBLOKI DS F SUM OF OUTPUT COUNT IDS 0420 COMMAND DS F SECONDARY ALLOCATION WORD 0420 COMMAND DS F SECONDARY ALLOCATION WORD 0424 COMMOND DS F SECONDARY ALLOCATION WORD 0428 COMMOND DS F CLOSE LIST AREA COMIDSNM EQU COMCLOSE INPUT DATA SET NAME ** PHASE LOCAL WORK AREA 042C COMLOCAL DS 22F ** MERGE/SORT INITIALIZATION ROUTINE 0484 COMERGEI DS 28F ** MERGE/SORT ROUTINE 0484 COMERGES DS 48F COMTABLE EQU COMERGES -4 TRANSLATE/SIZE TABLE ** TREE INITIALIZATION ROUTINE 0584 COMTREED DS 36F ** TREE ADD ROUTINE 0584 COMTREED DS 36F ** TREE ADD ROUTINE 0644 COMTREED DS 36F ** TREE DELETE ROUTINE 0646 COMTREED DS 16F	03D6	COMPCOPC DS	CL6	CLC 0(,R4),0(R14)		
03E8 COMPARE DS CL6 CLC 0(,R14),0(R15) 03E8 COMPSLAB DS CL6 CLC 4(,R4),4(R5) COMPSLAK EQU COMPCOPA COMPSLAK EQU COMPCOPB 03F4 COMPALTA DS CL6 CLC 4(,R2),0(R4) 03F4 COMPALTB DS CL6 CLC 4(,R3),0(R4) 0400 COMOVKAD DS CL18 MVC 0(KADLIST,R1),0(R15) ** BLOCK READ CHECK FIELDS ** BLOCK READ CHECK FIELDS 0418 DS 0D 0418 COMBLOKO DS F OUTPUT BLOCK COUNTER 041C COMBLOKI DS F INPUT BLOCK COUNTER 041C COMBLOKI DS F SUM OF OUTPUT COUNT IDS 0424 COMMAX DS F SUM OF OUTPUT COUNT IDS 0428 COMMAX DS F SECONDARY ALLOCATION WORD 0428 COMMAX DS F SECONDARY ALLOCATION WORD 0428 COMMAX DS F SECONDARY ALLOCATION WORD 0428 COMMONA DS F SECONDARY ALLOCATION WORD 0428 COMMONA DS F SECONDARY ALLOCATION WORD 0428 COMMONA DS F CLOSE LIST AREA COMIDSHM EQU COMCLOSE INPUT DATA SET NAME ** PHASE LOCAL WORK AREA 042C COMLOCAL DS 22F ** MERGE/SORT INITIALIZATION ROUTINE 0484 COMERGED DS 28F COMIDSHM EQU COMCLOSE INPUT DATA SET NAME ** TREE/MERGE ROUTINE SPACE ** MERGE/SORT ROUTINE 0484 COMERGES DS 48F COMTABLE EQU COMERGES -4 TRANSLATE/SIZE TABLE ** TREE INITIALIZATION ROUTINE 0584 COMTREED DS 36F ** TREE ADD ROUTINE 0584 COMTREED DS 36F ** TREE ADD ROUTINE 0684 COMTREED DS 16F	03DC	COMPOUND DS	CL6	CLC 0(,R2),0(R8)		
COMPSLAK EQU COMPCOPA *** *** *** *** *** ** ** **	U3E2	COMPTCK D2	CLO	CLC U(,R2),BVDRET-BVDSECT(R4)		
# BLUCK READ CHECK FIELDS 0418	0755	COMPCLAR DC	CL6 CL6 COMPCOPA	CLC 0(,R14),0(R15) CLC 4(,R4),4(R5)		
# BLUCK READ CHECK FIELDS **	03F4	COMPSEBE EQU	COMPCOPB	CLC 4(.82).0(84)		
# BLUCK READ CHECK FIELDS # BOUTPUT BLOCK COUNTER 0418 COMBLOKO DS F	03FA	COMPALTE DS	či 6	CLC 4(,R3),0(R4)		
0418 DS 0D 0418 COMBLOKO DS F OUTPUT BLOCK COUNTER 0410 COMBLOKI DS F INPUT BLOCK COUNTER 0420 COMHASH DS F SUM OF OUTPUT COUNT IDS 0424 COMZNDRY DS F SECONDARY ALLOCATION WORD 0428 COMNMAX DS F ESTIMATED NMAX VALUE ***********************************	0400	COMOVKAD DS	CF18			
## PHASE LOCAL WORK AREA **********************************		×	DECOR READ CHECK 1			
## PHASE LOCAL WORK AREA **********************************	0418	COMBLOKO DS	0D	CUITPUT RIGCY COUNTER		
## PHASE LOCAL WORK AREA **********************************	041C	COMBLOKI DS	F	INPUT BLOCK COUNTER		
## PHASE LOCAL WORK AREA **********************************	0420	COMHASH DS	F	SUM OF OUTPUT COUNT IDS		
## PHASE LOCAL WORK AREA **********************************	0428	COMMMAX DS	F	ESTIMATED NMAX VALUE		
042C COMLOCAL DS 22F * 0484 COMCLOSE DS 0F CLOSE LIST AREA COMIDSNM EQU COMCLOSE INPUT DATA SET NAME * TREE/MERGE ROUTINE SPACE * MERGE/SORT INITIALIZATION ROUTINE 0484 COMERGEI DS 28F * MERGE/SORT ROUTINE 04F4 COMERGES DS 48F COMTABLE EQU COMERGES-4 TRANSLATE/SIZE TABLE * TREE INITIALIZATION ROUTINE 05B4 COMTREEI DS 36F * TREE ADD ROUTINE 0644 COMTREEA DS 40F * TREE DELETE ROUTINE 06E4 COMTREED DS 16F		×	PHASE LOCAL WORK A	REA		
0484 COMCLOSE DS OF CLOSE LIST AREA COMIDSNM EQU COMCLOSE INPUT DATA SET NAME * TREE/MERGE ROUTINE SPACE * MERGE/SORT INITIALIZATION ROUTINE 0484 COMERGEI DS 28F * MERGE/SORT ROUTINE 04F4 COMERGES DS 48F COMTABLE EQU COMERGES-4 TRANSLATE/SIZE TABLE * TREE INITIALIZATION ROUTINE 05B4 COMTREEI DS 36F * TREE ADD ROUTINE 0644 COMTREEA DS 40F * TREE DELETE ROUTINE 06E4 COMTREED DS 16F	042C	COMLOCAL DS	22F			
X MERGE/SORT INITIALIZATION ROUTINE 0484 COMERGEI DS 28F	0484	COMCLOSE DS	OF COMCLOSE	INPUT DATA SET NAME		
* MERGE/SORT INITIALIZATION ROUTINE 0484 COMERGEI DS 28F * MERGE/SORT ROUTINE 04F4 COMERGES DS 48F COMTABLE EQU COMERGES-4 TRANSLATE/SIZE TABLE * TREE INITIALIZATION ROUTINE 05B4 COMTREEI DS 36F * TREE ADD ROUTINE 0644 COMTREEA DS 40F * TREE DELETE ROUTINE 06E4 COMTREED DS 16F		*	TREE/MERGE ROUTINE	SPACE		
* MERGE/SORT ROUTINE 04F4 COMERGES DS 48F COMTABLE EQU COMERGES-4 TRANSLATE/SIZE TABLE * TREE INITIALIZATION ROUTINE 05B4 COMTREEI DS 36F * TREE ADD ROUTINE 0644 COMTREEA DS 40F * TREE DELETE ROUTINE 06E4 COMTREED DS 16F		* MERGE/S	ORT INITIALIZATION			
04F4 COMERGES DS 48F	0484	COMERGEI DS MERGE/S	28F SORT ROUTINE	•		
05B4 COMTREEI DS 36F * TREE ADD ROUTINE 0644 COMTREEA DS 40F * TREE DELETE ROUTINE 06E4 COMTREED DS 16F	04F4	COMERGES DS COMTABLE EQU	48F COMERGES-4	TRANSLATE/SIZE TABLE		
* TREE DELETE ROUTINE 06E4 COMTREED DS 16F		COMTREEI DS * TREE AI	36F DD ROUTINE	_		
		* TREE DI	ELETE ROUTINE			
* TREE REPLACE/SELECT ROUTINE 0724 COM43LST DS OF LIST AREA FOR CON043A MSGS		* TREE R	EPLACE/SELECT ROUTIN			

```
DISPL NAME
                                                              DESCRIPTION
LAST LIST ENTRY
                  LONG COMPARE
                              DCL6
                                                              LONG COMPARE
07F0 COMPLONG DS
                  OFFSET LIST TO LAD FETCHES IN TREE CODE
07F0 COMPLADS DS
                               6F
                               VARIOUS WORK VARIABLES
       * DYNAMIC ALLOCATION PARAMETERS
O808 ORG COMTABLE+256
05F0 COMDYNAM DS CL8
05F8 COMDYNUM DS F
05FC COMVIO DS C

** ALTSEQ TRANSLATION TABLE
                               COMTABLE+256
                                                              SKIP OVER TABLE
                                                              DYNALLOC UNIT NAME
DYNALLOC COUNT
                                                               VIO/NOVIO SWITCH
05FD COMALTAB DS
                             CL256
                      ORG
06FD
                  E35 WORK AREAS
0808 COMOJECE DS
                             ```F
F
 ADDRESS OF SORTOUT JFCB BLOCKED LAD ADDR/ADDR(E35)
 LAST FOUR BYTES OF RECORD
0810 COMRTAIL DS
 USED FROM ICEDYNA UNTIL ECEEXIN
RETURN CODE FROM DYNALLOC
 SYSTEM RESERVED TRANSFER INDICATION FOR ICEMESS
 A=TRANSLATE AQ ONLY
C=TRANSLATE CH AND AQ
 Y=RECORD COUNTERS CHECKED
N=RECORD COUNTERS NOT CHECKED
'S' = SHORT SMF RECORD
'F' = FULL SMF RECORD
'F' NOT VALID FOR FLR-BLOCKSET
'N' = NO SMF RECORD
081E COMCHECK DS
 CL1
081F COMSMF DS
 C
0820 COMMXBLK DS
0824 COMTTIME DS
0828 COMMSGSW DS
COMB37R EQU
0829 COMVBLKS DS
082A COMBLKS DS
082B COMCNTL DS
082C COMTRACK DS
0830 COMAVGLN DS
 MAX SORTIN BLKSIZE FOR SMF
CPU TIME WORK AREA FOR SMF
INFORMATION MSG FLAG BYTE
B37 INFO MESSAGE FLAG MASK
N=BYPASS VLR BLOCKSET
N=BYBASS FLR BLOCKSET
 F
 .
X
X'80'
 C
 C
 X'00'=SORTCHTL NOT PRESENT
 #TRACKS ALLOCATED FOR SORTWK
AVERAGE RECORD LENGTH VALUE
 TRACE SEGMENTS
 1F TRACE X'01'-X'0F' E15/E35 IN/OUT AREA RECOR LEVEL
8F TRACE X'10'-X'1F' BLOCK LEVEL ENTRIES
8F TRACE X'20'-X'9F' BLOCKSET LEVEL ENTRIES
15F TRACE X'40'-X'FF' MODULE AND MAIN LOOPS
0834 COMTRACE DS
 DS
0838
0858
 DS
0878
 D5
 COMTRACE
08B4
 ORG
0834 COMOPLOC DS
 20F
 OPTION STATEMENT WORK AREA
0884
 ORG
08B4 COMRSAV1 DS F
08B8 COMRSAV2 DS F
08BC COMMSGPR DS H
 MORE SAVE AREAS
 REGISTER SAVE AREA
 REGISTER AREA
 MESSAGE PARAMETER SAVE AREA
```

Licensed Material--Property of IBM
Page of LY33-8042-6, As Updated 31 March 1981, By TNL LN20-9345

DISPL	NAME			DESCRIPTION
	*		VLR FIELD AREA	× × 
08C0 08C4 08C8 08C8 08D0 08D8 08D8 08E0 08E0 08E6 08F6 08F6 0904 0904 0904 0904 0914 0916	COMIRES COMWRES COMWRES COMMESIZ COMFLOOR COMMINSEG COMMINSEG COMMINSEG COMMINSEG COMMINSTOR COMMIN	DS DS DS DS DS DS DS DS DS DS DS DS DS D	<b>FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF</b>	START OF RESIDUAL INPUT DATA START OF RESIDUAL WORK DATA START OF RESIDUAL OUTPUT DATA SIZE OF WORK AREA PAGE SIZE COMPLEMENT RESIDUE DELTA MIN SIZE FOR 1ST SEGMENT WORK BUF TO RSA ROUTINE RSA TO WORK BUF ROUTINE RESTORE LIST POINTER FIXED RESTORE ROUTINE ADDRESS PER CENT RSA UTILIZATION ADDRESS OF UNBALANCED MERGE UNBALANCED LIST HEAD UNBALANCED LIST HEAD UNBALANCED LIST END PEERAGE ACCUMULATOR POINTER TO LOW LIMIT KEY AREA TOTAL RSA BYTES NOT IN OUTPUT RUN OFFSET IN BIN TO SEG BYTE BACKOUT RECORD'S PREDECESSOR TOTAL BYTES SEGMENTED BYTES IN WORK FILE SORTIN BYTE COUNT
0954 0958 095C	COMESAVE COMECTRS COMKSAVE COMLSTAT COMLSTAB COMEXTRA	DS DS DS	3F 10F F F F F 2F	EVENT COUNT SAVE AREA EVENT COUNT SAVE AREA KEY SAVE AREA ADDRESS LENGTH STATISTICS CTR PTR LENGTH STATISTICS TABLE UNUSED WORDS
0968	* COMFEND COMMVARY			END OF FOUNDATION BEGINNING OF VARIABLE AREA

## Peerage and Vale Communication Area (COMMA)

- | This area is allocated by ICEDEF and is used by all Peerage and Vale technique modules. It is described in the DSECT ICECOMMA. The program uses register 13 as a base register for COMMA.
- | It can be listed as described in Section 6, 'Debugging Aids'. Such a listing gives not only field definitions (name, position, length, description), but also current contents.

DISP	LEVEL	LABEL	ATTR	COMMENT
000	2	CSAVEOS	PTR(31)	SUPERVISOR AND DM SAVE AREA
048 090	2 2 2 3	CSAVEL1 CSAVEL2	PTR(31)	LEVEL 1 ROUTINE SAVE AREA LEVEL 2 ROUTINE SAVE AREA
0 D8	2	CSAVEL3		LEVEL 3 ROUTINE SAVE AREA
120 120	2,	CTEMP1	FIXED(31) FIXED(31)	WORK AREA Work Area
121	3	CWORK1 CTEMP124	PTR(24)	WORK AREA
121	5	CWORK124	PTR(24)	WORK AREA
122 122	6	CTEMP115	FIXED(15)	WORK AREA
123	<b>'</b> 8	CWORK116 CTEMP108	FIXED(16) PTR(8)	WORK AREA Work Area
123	ັ 9	CWORK108	PTR(8)	WORK AREA
124	2 7	CTEMP2	FIXED(31)	WORK AREA
124 125	3,	CWORK2 CTEMP224	FIXED(31) PTR(24)	WORK AREA Work Area
125	ັ5	CWORK224	PTR(24)	WORK AREA
126	6_	CTEMP215	FIXED(15)	WORK AREA
126 127	, 8	CWORK216 CTEMP208	FIXED(16) PTR(8)	WORK AREA Work Area
127	<b>5</b> 9	CWORK208	PTR(8)	WORK AREA
128	2	CTEMP3	FIXED(31)	WORK AREA
128 129	ა ნ	CWORK3 CTEMP324	FIXED(31) PTR(24)	WORK AREA Work Area
129	ິ6	CWORK324	PTR(24)	WORK AREA
12A	7	CTEMP315	FIXED(15)	WORK AREA
12A 12B	8	CWORK316 CTEMP308	FIXED(16) PTR(8)	WORK AREA Work Area
12B	10	CWORK308	PTR(8)	WORK AREA
12C	2 3	CTEMP4	FIXED(31)	WORK AREA
12C 12D	3	CWORK4 CTEMP424	FIXED(31) PTR(24)	WORK AREA WORK AREA
12D	ั้ 5	CWORK424	PTR(24)	WORK AREA
12E	6_	CTEMP415	FIXED(15)	WORK AREA
12E 12F	7 8	CWORK416 CTEMP408	FIXED(16) PTR(8)	WORK AREA WORK AREA
12F	ັງ	CWORK408	PTR(8)	WORK AREA
130	2	CWORK5	FIXED(31)	WORK AREA
134 138	2	CWORK6 CWORK7	FIXED(31) FIXED(31)	WORK AREA WORK AREA
13C	2	CWORK8	FIXED(31)	WORK AREA
140	2_	CRTNADDR		ADDR. OF PERMANENT RTHS
140 144	3	CGETMAIN CFRESTOR	PTR(31) PTR(31)	ADDR. OF ROUTINE GETMAINS ADDR. OF ROUTINE FREESTOR
148	222223333333333333333333333333333333333	CGALADDR	PTR(31)	ADDR. OF ROUTINE GALLOCAT
14C	3	CRALADDR	PTR(31)	ADDR. OF ROUTINE RALLOCAT
150 154	3 3	CDALADDR CBALADDR	PTR(31) PTR(31)	ADDR. OF ROUTINE DALLOCAT ADDR. OF ROUTINE BALLOCAT
158	3	CTRKROUT	PTR(31)	ADDR. OF ROUTINE TRKROUT
15C	3	CIORADDR	PTR(31)	ADDR. OF ROUTINE INOUTRIN
160 164	3 3	CMSGADDR Ctime	PTR(31) PTR(31)	ADDR. OF ROUTINE DIAGNOSE ADDR. OF ROUTINE TIMING
168	ž	CRECHAIN	PTR(31)	ADDR. OF ROUTINE RECHAIN
16C	3	CSTAADDR	PTR(31)	ADDR OF INIT STAE

CXLIST PTR(31) ADDRESS OF EXIT LIST COMMODS FIXED(31) COREPARM SAVE FOR ICEDEV DOUBLE DOYCOMORD DOUBLEWORD—ALIGNED WORKAREA CXCCOUNT FIXED(31) WORK AREA FOR EXTRACT CEQUISE PTR(31) WORK AREA FOR CEXTRACT CEQUISE PTR(31) WORK AREA FOR CEXTRACT CEQUISE PTR(31) SIZE OF ZEROTH FREE AREA CEQUISE PTR(31) SIZE OF ZEROTH FREE AREA CEQUISE PTR(31) ADDR. OF TREEADDS ROUTINE CEQUISE PTR(31) ADDR. OF TREEADDS ADDR. OF TREEAD	DISP	LEVEL	LABEL	ATTR	COMMENT
5 CEQDISPL PIR(31) MURK AREA FOR MOVECOMP  4 CMOVEH PIR(31) MORK AREA FOR EXTRACT  4 CMOVEH PIR(31) MORK AREA FOR EXTRACT  5 CFRELIST BDY(DWORD) START OF FREE SPACE CHAIN  3 CFREHEAD PIR(31) ADDR. OF FIRST FREE AREA  2 CADD PIR(31) SIZE OF ZEROTH FREE AREA  3 CADDISP FIXED(31) TREEGEN WORK CONSTANT  2 CADDS PIR(31) ADDR. OF TREEADD ROUTINE  3 CINSTRIT FIXED(31) TREEGEN WORK CONSTANT  2 CADDS PIR(31) ADDR. OF TREEADDS ROUTINE  3 CINSTRIT FIXED(31) TREEGEN WORK AREA  2 CBINBLK PIR(31) ADDR. OF TREEADDS ROUTINE  2 CBASEMIN FIXED(31) TREEGEN WORK AREA  2 CBINBLK PIR(31) ADDR. OF TREEADDS ROUTINE  2 CBINSIZE PIR(31) RSA BIN SIZE FOR MERGING  2 CBINSIZE PIR(31) ADDR. OF TREEADDS ROUTINE  2 CBLKITIR PIR(31) ADDR. OF TREEADD ROUTINE  2 CBLKISIZ FIXED(31) TREEGEN WORK AREA  2 CBLKISIZ FIXED(31) ADDR. OF TREEADD ROUTINE  3 * PIR(31) ADDR. OF TREEADD ROUTINE  4 CREAD AND AND AND AND AND AND AND AND AND A		_			
TOUVERNY PIRCISI) MORK AREA FOR MOVECOMP OF CEDITSPL PIRCISI) MORK AREA FOR EXTRACT OF CEDITSPL PIRCISI) MORK AREA FOR EXTRACT OF COMPANY OF CO	3	2			ADDRESS OF EXIT LIST
4 CHOVEEND PIR(31) MURK AREA FOR MOVECOMP  3 CROUVER PIR(31) MORK AREA FOR EXTRACT  4 CMOVEN PIR(31) MORK AREA FOR EXTRACT  3 CFREHEAD PR(31) MORK AREA FOR EXTRACT  3 CFREHEAD PR(31) MORK AREA FOR EXTRACT  4 CMOVEN PIR(31) MORK AREA FOR MOVECOMP  5 CFREERO FIXED(31) START OF FREE SPACE CHAEN  3 CFREERO FIXED(31) SIZE OF ZEROTH FREE AREA  4 CADD PR(31) ADDR. OF TREEADD ROUTINE  5 CADDS PIXED(31) TREEGEN MORK CONSTANT  2 CADDS PIXED(31) TREEGEN MORK CONSTANT  2 CADSENIN FIXED(31) ADDR. OF TREEADDS ROUTINE  2 CADSENIN FIXED(31) TREEGEN MORK AREA  2 CABSENIN FIXED(31) ADDR. OF TREEADDS ROUTINE  2 CABSENIN FIXED(31) ADDR. OF TREEADDS ROUTINE  2 CABSENIN FIXED(31) TREEGEN MORK AREA  2 CABSENIN FIXED(31) TREEGEN MORK AREA  2 CABSENIN FIXED(31) ADDR. OF TREEADDS ROUTINE  2 CABSENIN FIXED(31) ADDR. OF TREEADDS ROUTINE  2 CABSENIN FIXED(31) TREEGEN MORK AREA  2 CABSENIN FIXED(31) ADDR. OF TREEADDS ROUTINE  2 CABSENIN FIXED(31) ADDR. OF CURRENT MORK BUF  2 CABSENIN FIXED(31) ADDR. OF CURRENT MORK BUF  2 CABSENIN FIXED(31) ADDR. OF CURRENT MORK BUF  2 CABSENIN FIXED(31) INDEX AREA 1 HEAD  2 CCHIDX1 FIXED(31) INDEX AREA 2 HEAD  2 CCHIDX2 FIXED(31) INDEX AREA 2 HEAD  2 CCHIDX3 FIXED(31) INDEX AREA 2 HEAD  2 CCHIDX4 FIXED(31) ADDR. OF NORTH PARCEL FIELDS  4 CIONALS FIXED(31) CONSTANT = 256  2 CCONNEG4 FIXED(31) CONSTANT = 256  2 CCONNEG4 FIXED(31) CONSTANT = 256  2 CCORESAV FIXED(31) ADDR. OF INIT ROUT  3 CIOSAVI FIXED(31) ADDR. OF NORTH PARCEL FIELDS  4 CIOSAVI FIXED(31) ADDR. OF SORTIN/OUT DECB 1EC  2 COCHERER FIXED(31) ADDR. OF SORTIN/OUT DECB 1EC  2 COCHERE FIXED(31) ADDR. OF FREEE BUFFER  3 COPHANC PIR(31) ADDR. OF TREEGEN BR-TABLE  4 COCHERE FIXED(31) ADDR. OF TREEGEN BR-TABLE  5 COEXTRE FIXED(31) ADDR. OF TREELETE ROUTINE  3 COPHANC PIR(31) ADDR. OF SORTIN/OUT DECB 1EC  4 COEXTRE FIXED(31) ADDR. OF TREELETE ROUTINE  5 COEXTRE FIXED(31) ADDR. OF TREELETE ROUTINE  5 COEXTRE FIXED(31) ADDR. OF TREEGEN BR-TABLE  COEXTRE FIXED(31) ADDR. OF TREELETE ROUTINE  2 COEXTRE FIXED(31) ADDR. OF TREELETE ROUTINE  3 COEXTRE FIXED(31)		2			CUREPARM SAVE FOR ICEDEV
SCEODISPL PIR(31) WORK AREA FOR EXTRACT  4 CMOVEH PIR(31) WORK AREA FOR EXTRACT  5 CEGOISPL PIXED(31) WORK AREA FOR EXTRACT  6 CMOVEH PIR(31) WORK AREA FOR EXTRACT  7 CFRELIST BDY(DWORD) START OF FREE SPACE CHAPA  8 CFREZERO FIXED(31) SIZE OF FIRES PACE CHAPA  8 CFREZERO FIXED(31) SIZE OF FIRES PACE CHAPA  9 CADDD PIR(31) ADDR. OF TREEADD ROUTINE  10 CADDS PIXED(31) ADDR. OF TREEADD ROUTINE  11 CADDS PIXED(31) ADDR. OF TREEADD ROUTINE  12 CADDS PIXED(31) TREEGEN WORK CONSTANT  12 CADSSEMIN FIXED(15) BASSE-BIN MIN ALLOC  12 CBINBLK PIR(31) ADDR. OF TREEADDS ROUTINE  12 CBINSIZE PIR(31) ADDR. OF TREEADDS ROUTINE  13 CINSTRIT FIXED(31) ADDR. OF TREEADDS ROUTINE  14 CBINSIZE PIR(31) ADDR. OF TREEADDS ROUTINE  15 CBINSIZE PIR(31) ADDR. OF TREEADDS ROUTINE  16 CBINSIZE PIR(31) ADDR. OF TREEADDS ROUTINE  17 CBINSIZE PIR(31) ADDR. OF TREEADDS ROUTINE  18 CBINSIZE PIR(31) ADDR. OF CURRENT WORK BUF  19 CBINSIZE PIR(31) ADDR. OF CURRENT WORK BUF  20 CBINSIZE PIR(31) ADDR. OF CURRENT WORK BUF  21 CBINSIZE PIR(31) ADDR. OF AVAIL VIR BUFF ADDR  22 CCHIDX1 FIXED(31) INDEX AREA 1 HEAD  23 CCHIDX2 FIXED(31) INDEX AREA 2 HEAD  24 CCHIDX2 FIXED(31) INDEX AREA 2 HEAD  25 CCHIDX3 FIXED(31) ADDR. OF ANTHON AREA  26 CCHIDX5 FIXED(31) CONSTANT = -4  27 CCHIDX5 FIXED(31) CONSTANT = -4  28 CCHIDX5 FIXED(31) CONSTANT = -4  29 CCHIDX6 FIXED(31) ADDR. OF INIT ROUT  20 CDEATED PIR(31) ADDR. OF INIT ROUT  21 CDEATED PIR(31) ADDR. OF INIT ROUT  22 CDEATED PIR(31) ADDR. OF INIT ROUT  23 CIGNAL FIXED(31) ADDR. OF INIT ROUT  24 CICONACA PIR(31) ADDR. OF INIT ROUT  25 CDECAREA PIR(31) ADDR. OF INIT ROUT  26 CDEATED PIR(31) ADDR. OF TREEGEN BR-TABLE  27 COURSESAV FIXED(31) ADDR. OF SORTIN/OUT DECB 1EC  28 CDECAREA PIR(31) ADDR. OF SORTIN/OUT DECB 1EC  29 CDECAREA PIR(31) ADDR. OF TREEGEN BR-TABLE  20 CDECAREA PIR(31) ADDR. OF TREELETE ROUTINE  21 CDECAREA PIR(31) ADDR. OF TREELETE ROUTINE  22 CDELETE PIR(31) ADDR. OF TREELETE ROUTINE  23 COUNNER PIR(31) ADDR. OF TREELETE ROUTINE  24 CDEXTR FIXED(31) ADDR. OF TREELETE ROUTINE  25 CDECARES PIR(31) A		۷,			DOUBLEWORD-ALIGNED WORKAREA
CXCLNG FIXED(31) WORK AREA FOR EXTRACT WORKOWSH PT(31) WORK AREA FOR MOVECOMP START OF CREET STATE CHAIN ADDR. OF FIRST FREE AREA STATE OF FREE SPACE CHAIN ADDR. OF FIRST FREE AREA STATE CHAIN ADDR. OF FIRST FREE AREA STATE CHAIN ADDR. OF FIRST FREE AREA STATE CHAIN ADDR. OF TREEADD ROUTINE ADDR. OF TREEADDS ROUTINE ADDR. OF TREEADD		٠ ,			
CXCLNG FIXED(31) WORK AREA FOR EXTRACT WORK CMOVECOMP START OF TREE SPACE CHAIN AS CFRELEAD PTR(31) START OF TREE SPACE CHAIN ADDR. OF FIRST FREE AREA START OF TREE SPACE CHAIN ADDR. OF FIRST FREE AREA START OF TREE SPACE CHAIN ADDR. OF FIRST FREE AREA START OF TREE SPACE CHAIN ADDR. OF FIRST FREE AREA START OF TREE SPACE CHAIN ADDR. OF FIRST FREE AREA START OF TREE SPACE CHAIN ADDR. OF FIRST FREE AREA START OF TREE SPACE CHAIN ADDR. OF FIRST FREE AREA START OF TREE SPACE CHAIN ADDR. OF TREEADD ROUTINE START OF TREE SPACE CHAIN ADDR. OF TREEADD ROUTINE START OF TREE SPACE CHAIN ADDR. OF TREEADD ROUTINE START OF TREE SPACE CHAIN ADDR. OF TREEADD ROUTINE START OF TREE SPACE CHAIN ADDR. OF TREEADD ROUTINE START OF TREE SPACE CHAIN ADDR. OF TREEADD ROUTINE START OF TREE SPACE CHAIN ADDR. OF TREEADD ROUTINE START OF TREE SPACE CHAIN ADDR. OF TREEADD ROUTINE START OF TREE SPACE CHAIN ADDR. OF TREEADD ROUTINE START OF TREE SPACE CHAIN ADDR. OF TREEADD ROUTINE START OF TREE SPACE CHAIN ADDR. OF TREEADD ROUTINE START OF TREE SPACE CHAIN ADDR. OF TREEADD ROUTINE START OF TREE SPACE CHAIN ADDR. OF TREEADD ROUTINE START OF TREE SPACE CHAIN ADDR. OF TREEADD ROUTINE START OF TREE SPACE CHAIN ADDR. OF TREEADD ROUTINE START OF TREE SPACE CHAIN ADDR. OF TREEADD ROUTINE START OF TREE SPACE CHAIN ADDR. OF TREE SHOCK LIST OUT. ADDR. OF TREE SHOCK LIST OF TREE SHOCK LIST OF TREE SHOCK LIST OUT. ADDR. OF TREE SHOCK LIST OF TREE SHOCK LIST OUT. ADDR. OF TREE SHOCK LIST OUT. ADDR. OF TREE SHOCK LIST OUT. ADDR. OF STATT OUT. DECR. SHOW ADDR. OF STATT OUT. D	3 3	75			WORK AREA FOR MUVECUMP
4 CMOVEH PIR(31) WORK AREA FOR MOVECOMP 2 CFREHEAD PIR(31) ADDR. OF FIRES PACE CHAIN 3 CFREZERO FIXED(31) SIZE OF ZEROTH FREE AREA 2 CADD PIR(31) ADDR. OF TREEADD ROUTINE 3 CADDISP FIXED(31) TREEGEN WORK CONSTANT 2 CADDS PIR(31) ADDR. OF TREEADDS ROUTINE 3 CADDS PIR(31) ADDR. OF TREEADDS ROUTINE 2 CADS PIR(31) ADDR. OF TREEADDS ROUTINE 3 CINSTRIT FIXED(31) TREEGEN WORK CONSTANT 2 CBASEMIN FIXED(15) BASE-BIN MIN ALLOC 2 CBINBLK PIR(31) ADDR TO BINBLK Q 2 CBINSIZE PIR(31) ADDR TO BINBLK Q 2 CBINSIZE PIR(31) ADDR TO BINBLK Q 2 CBINSIZE PIR(31) NEXT UNUSED ITTR 2 CBUFFER PIR(31) ADDR TO BINBLK Q 2 CBLKISIZ FIXED(31) SIZE OF FREE BLOCK LIST 3 ** PIR(31) OUT - AVAIL VLR BUFF ADDR 2 CCHIDX1 FIXED(31) INDEX AREA 1 HEAD 2 CCHIDX2 FIXED(31) INDEX AREA 2 HEAD 2 CCHIDX3 FIXED(31) INDEX AREA 1 HEAD 2 CCHIDX4 FIXED(31) INDEX AREA 2 HEAD 2 CCHIDX5 FIXED(31) POOL AREA 1 HEAD 2 CCHIDX5 FIXED(31) POOL AREA 2 HEAD 3 CIONAC66 FIXED(31) POOL AREA 2 HEAD 4 CIONAC66 FIXED(31) CONSTANT = 256 2 CCORRESAV FIXED(31) INDEX AREA 2 HEAD 4 CIONAC6 FIXED(31) ORIGINAL SIZE PARM 4 CIONATT FIXED(31) ADDR OF INIT ROUT 3 CIONAC64 FIXED(31) ADDR OF INIT ROUT 4 CIONATT FIXED(31) ADDR OF INIT ROUT 5 COCRESAV FIXED(31) TO ASK SAVE AREA 4 CIONATT FIXED(31) ADDR OF OPEN-CLOSE LIST 4 CIONATT FIXED(31) ADDR OF OPEN-CLOSE LIST 5 CDELETE PIR(31) ADDR OF TREELETE ROUTINE 5 CDELETE PIR(31) ADDR OF TREELETE ROUTINE 6 COCRESAV FIXED(31) ADDR OF TREELETE ROUTINE 7 COCRESAV FIXED(31) ADDR OF TREELETE ROUTINE 8 COCRESAV FIXED(31) ADDR OF TREELETE ROUTINE 9 COCRESAV FIXED(31) ADDR OF TREELETE ROUTINE 1 COCRESAV FIXED(31) ADDR OF TREELETE ROUTINE 2 COCRESAV FIXED(31) ADDR OF TREELETE ROUTINE 2 COCRESAV FIXED(31) ADDR OF TREELETE ROUTINE 2 COCRESAV FIXED(31) ADDR OF TREELETE ROUTINE 3 COCRESAV FIXED(31) ADDR OF TREELETE ROUTINE 2 COCRESAV FI		3	CVCLNC	CIVED/711	HORE AREA FOR EXTRACT
CFREHEAD PTR(31) ADDR. OF FIRST FREE AREA CADD PTR(31) ADDR. OF TREEADD ROUTINE CADDS PTXED(31) TREEGEN WORK CONSTANT CADDS PTX(31) ADDR. OF TREEADDS ROUTINE CADDS PTXED(31) TREEGEN WORK CONSTANT CADDS PTXED(31) ADDR. OF TREEADDS ROUTINE CADDS PTXED(31) ADDR. OF TREEADDS ROUTINE CADDS PTX(31) ADDR. OF TREEADCS LIST CADDA AREA 1 HEAD CADDS PTX(31) ADDR. OF TREEADCS LIST CADDA AREA 1 HEAD CADDS PTX(31) ADDR. OF TREEADCS LIST CADDA AREA 1 HEAD CADDA AREA 1 HEAD CADDA AREA 1 HEAD CADDA AREA 2 HEAD CADDA AREA 3		4	CMOVEH	PTR(31)	
CADD CADD TR(SI) ADDR. OF TREEADDR ROUTINE CADDS CADDS CADDS CADDS TR(SI) TREEGEN WORK CONSTANT TREEADDR ROUTINE CADDS CADDS CADDS TR(SI) TREEGEN WORK CONSTANT TREEADDR ROUTINE CADSARMIN FIXED(SI) TREEGEN WORK AREA CADSARMIN FIXED(SI) ADDR. OF TREEADDR ROUTINE CADSARMIN ALLOC CADINSIZE CADIN		2	CFRELIST	BDY(DWORD)	START OF FREE SPACE CHAIN
CADD CADD TR(SI) ADDR. OF TREEADDR ROUTINE CADDS CADDS CADDS CADDS TR(SI) TREEGEN WORK CONSTANT TREEADDR ROUTINE CADDS CADDS CADDS TR(SI) TREEGEN WORK CONSTANT TREEADDR ROUTINE CADSARMIN FIXED(SI) TREEGEN WORK AREA CADSARMIN FIXED(SI) ADDR. OF TREEADDR ROUTINE CADSARMIN ALLOC CADINSIZE CADIN	)	_3		PTR(31)	ADDR. OF FIRST FREE AREA
2 CADD PTR(31) ADDR. OF TREEADD ROUTINE 3 CADDS PIXED(31) TREEGEN WORK CONSTANT 2 CADS PTR(31) ADDR. OF TREEADDS AROUTINE 3 CINSTRIT FIXED(31) TREEGEN WORK AREA 2 CBASEMIN FIXED(31) TREEGEN WORK AREA 2 CBINBLK PTR(31) ADDR. OF TREEADDS AROUTINE 2 CBINBLK PTR(31) ADDR. TO BINBLK Q 2 CBINBLK PTR(31) RSA BIN SIZE FOR MERGING 2 CBLKITTR PTR(31) ADDR. OF CURRENT WORK BUF 3 CBLKSIZ FIXED(31) SIZE OF FREE BLOCK LIST 2 CBLKSIZ FIXED(31) SIZE OF FREE BLOCK LIST 3 PTR(31) OUT - AVAIL VUR BUFF ADDR 2 CCHIDX1 FIXED(31) INDEX AREA 1 HEAD 3 CCHIDX2 FIXED(31) INDEX AREA 2 HEAD 4 CCHIDX2 FIXED(31) INDEX AREA 2 HEAD 5 CCHIDX3 FIXED(31) INDEX AREA 2 HEAD 6 CCHIDX4 FIXED(31) POOL AREA 1 HEAD 7 CCONNEG4 FIXED(31) CONSTANT = 26 7 CCONNEG4 FIXED(31) CONSTANT = 26 8 CCONNEG4 FIXED(31) CONSTANT = 26 8 CCONSES FIXED(31) ORIGINAL SIZE PARM 8 DY(WORD) I/O INTERFACE FIELDS 9 CCORESAV FIXED(31) ADDR OF INIT ROUT 1 ADDR. OF OPPHYLOLOGE LIST 1 CODSAVI3 FIXED(31) ADDR OF INIT ROUT 2 COLOBLIST PTR(31) ADDR OF INIT ROUT 3 CIOSAVI3 FIXED(31) ADDR OF INIT ROUT 4 CICGRERA PTR(31) ADDR. OF OPPHYLOLOGE LIST 9 CDELETE FIXED(31) ADDR. OF OPPHYLOLOGE LIST 1 CDATLOC BDY(WORD) ICEMAN BASEREGS 2 CDATLOC BDY(WORD) ICEMAN BASEREGS 2 CDATLOC BDY(WORD) ICEMAN BASEREGS 3 CDATLOR BDY(WORD) ICEMAN BASEREGS 4 CDECERE PTR(31) ADDR. OF TREELETE ROUTINE 5 CDELETE FIXED(31) ADDR. OF TREELETE ROUTINE 6 CDECERE PTR(31) ADDR. OF TREELETE ROUTINE 7 CDECERE PTR(31) ADDR. OF TREELETE ROUTINE 8 CDECERE PTR(31) ADDR. OF TREELETE ROUTINE 9 CDEXITTR PTR(31) ADDR. OF TREELETE ROUTINE 1 ADDR. OF TREELETE ROUTINE 1 ADDR. OF TREELETE ROUTINE 1 ADDR. OF TREELETE PROUTINE 2 CDEXITR PTR(31) ADDR. OF TREELETE ROUTINE 2 CDEXITR PTR(31) ADDR. OF TREELETE PROUTINE 3 CONSTANT TO THE PTR TO THE TREELET PROUTINE 4 CDEXTRACT PTR(31) ADDR. OF TREELET PROUTINE 5 CDEVAIN PTR(8) TREELET PROUTINE 6 CDEVAIN PTR(8) TREELET PROUTINE 7 CDEVAIN PTR(8) TREELET PROUTINE 8 CONSTANT TO THE PTR TO THE TREELET PROUTINE 9 COUNTS TO THE TREELET PROUTINE 1 COUNTS TO THE TREELET PROUTINE 1 COUNTS TO THE TREELET	4	3			SIZE OF ZEROTH FREE AREA
2 CADDS PTR(31) ADDR. OF TREEADDS ROUTINE 2 CBASEMIN FIXED(15) BASE-BIN MIN ALLOC 2 CBINSIZE PTR(31) ADDR. OF TORRENDE MORK AREA 2 CBINSIZE PTR(31) ADDR. TO BINBLK Q 2 CBINSIZE PTR(31) ADDR. OF CURRENT WORK BUF 3 CBLKITIR PTR(31) ADDR. OF CURRENT WORK BUF 4 CBLIDX1 FIXED(31) SIZE OF FREE BLOCK LIST 5 CBLKLSIZ FIXED(31) SIZE OF FREE BLOCK LIST 6 CBLKLSIZ FIXED(31) IN PTR IN BUFF TO VLR 7 CCHIDX2 FIXED(31) INDEX AREA 1 HEAD 2 CCHIDX2 FIXED(31) INDEX AREA 1 HEAD 3 CCHIDX3 FIXED(31) INDEX AREA 2 HEAD 4 CCHIDX3 FIXED(31) POOL AREA 1 HEAD 5 CCHIDX4 FIXED(31) POOL AREA 1 HEAD 6 CCONNEG4 FIXED(31) CONSTANT = -56 6 CCONNEG4 FIXED(31) CONSTANT = -56 7 CCONNEG4 FIXED(31) ORIGINAL SIZE PARM 8 DY(MORD) I/O INTERFACE FIELDS 8 DY(MORD) I/O INTERFACE FIELDS 9 COCORSSAV FIXED(31) ADDR OF INIT ROUT 1 COLONIAL FIXED(31) ADDR OF INIT ROUT 2 CCURRBLK PTR(31) ADDR OF INIT ROUT 3 CIDSAVI3 FIXED(31) ADDR OF INIT ROUT 4 CTOINIT FIXED(31) ADDR OF OF INIT ROUT 5 CODELIST PIXED(31) ADDR OF FORM/CLOSE LIST 6 CDATLOC BDY(MORD) I/O TASK SAVE AREA 7 CODELIST PTR(31) ADDR OF OF ORM/CLOSE LIST 8 CDELETE PTR(31) ADDR OF OF ORM/CLOSE LIST 9 CDELETE PTR(31) ADDR OF OF ORM/CLOSE LIST 1 CDELETE PTR(31) ADDR OF OF ORM/CLOSE LIST 1 CDECAREA PTR(31) ADDR OF OF TRELETE ROUTINE ADDR OF TRELETE PER BUFFER 1 CDEXTITE PTR(31) ADDR OF TRELETE PER BUFFER 2 CDEXITTE PTR(31) ADDR OF TRELETE PER BUFFER 3 CONNAIL PTR(3) ADDR OF UNUSED DYNAMIC AREA 4 CONNAIL PTR(3) ADDR OF TRELETE PER BUFFER 5 CDEXITE PTR(31) ADDR OF TRELETE PER BUFFER 6 CDEXTRE FIXED(31) INDEX ENTRIES PER BUFFER TRACK 6 CDYNAUK CHAR(8) DYN ALLOC DEVICE TYPE 7 CONNAIL PTR(8) TREED(31) TO THE PER TO THE TRACK 8 CONNAIL PTR(8) TREED(31) TO THE PER TO THE TRACK 9 CONNAIL PTR(8) TREED(31) TO THE PER TO THE TRACK 9 CONNAIL PTR(8) TREED(31) TO THE PER TO THE PER SUPPLIES 9 CONNAIL PTR(8) TREED(31) TO THE PER SUPPLI	3	2		PTR(31)	ADDR. OF TREEADD ROUTINE
CBASEMIN FIXED(15) BASE-BIN MIN ALLOC CBINBIK PTR(31) ADDR TO BINBLK Q CBINSIZE PTR(31) ADDR TO BINBLK Q CBLKITTR PTR(31) NEXT UNUSED ITTR CBLKITTR PTR(31) NEXT UNUSED ITTR CBLKISIZ FIXED(31) SIZE OF FREE BLOCK LIST CBLKLSIZ FIXED(31) IN - PTR IN BUFF TO VLR CBLKLSIZ FIXED(31) INDEX AREA 1 HEAD CCHIDX1 FIXED(31) INDEX AREA 2 HEAD CCHIDX2 FIXED(31) INDEX AREA 2 HEAD CCHIDX3 FIXED(31) INDEX AREA 2 HEAD CCHIDX3 FIXED(31) POOL AREA 1 HEAD CCHIDX4 FIXED(31) POOL AREA 1 HEAD CCHIDX5 FIXED(31) POOL AREA 2 HEAD CCHIDX5 FIXED(31) CONSTANT = 256 CCHIDX6 FIXED(31) CONSTANT = 256 CCONES64 FIXED(31) TO RIGINAL SIZE PARM CCURBAL FIXED(31) TO RIGINAL SIZE PARM CCURBAL FIXED(31) TO TASK SAVE AREA CIDSAVI3 FIXED(31) TO TASK SAVE AREA CIDSAVI3 FIXED(31) ADDR OF INIT ROUT CCHORD FIXED(31) ADDR OF UNIT ROUT CCHORD FIXED(31) ADDR TO CURR BINBLK CCURBAL FIXED(31) ADDR TO CURR BINBLK CCURBAL FIXED(31) ADDR TO CURR BINBLK CCUCRBL FIXED(31) ADDR TO CURR BINBLK CCURBAL FIXED(31) ADDR TO CURR BINBLK CCURBAL FIXED(31) ADDR OF OSTIN-OUT DECB 1EC CDATLOR FIXED(31) ADDR OF OSTIN-OUT DECB 1EC CDECCTR FIXED(31) ADDR OF OSTIN-OUT DECB 1EC CDECCTR FIXED(31) ADDR OF OSTIN-OUT DECB 1EC CDECCTR FIXED(31) ADDR OF OSTIN-OUT DECB 1EC CDEXCTR FIXED(31) ADDR OF OSTIN-OUT DECB 1EC CDESUG CHAR(1) DEBUG SMITCH BYTE CCEMPBUDE CHAR(1) DEBUG SMITCH BYTE CCEMPBUDE	В	_ 3		FIXED(31)	TREEGEN WORK CONSTANT
CBASEMIN FIXED(15) BASE-BIN MIN ALLOC CBINBIK PTR(31) ADDR TO BINBLK Q CBINSIZE PTR(31) ADDR TO BINBLK Q CBLKITTR PTR(31) NEXT UNUSED ITTR CBLKITTR PTR(31) NEXT UNUSED ITTR CBLKISIZ FIXED(31) SIZE OF FREE BLOCK LIST CBLKLSIZ FIXED(31) IN - PTR IN BUFF TO VLR CBLKLSIZ FIXED(31) INDEX AREA 1 HEAD CCHIDX1 FIXED(31) INDEX AREA 2 HEAD CCHIDX2 FIXED(31) INDEX AREA 2 HEAD CCHIDX3 FIXED(31) INDEX AREA 2 HEAD CCHIDX3 FIXED(31) POOL AREA 1 HEAD CCHIDX4 FIXED(31) POOL AREA 1 HEAD CCHIDX5 FIXED(31) POOL AREA 2 HEAD CCHIDX5 FIXED(31) CONSTANT = 256 CCHIDX6 FIXED(31) CONSTANT = 256 CCONES64 FIXED(31) TO RIGINAL SIZE PARM CCURBAL FIXED(31) TO RIGINAL SIZE PARM CCURBAL FIXED(31) TO TASK SAVE AREA CIDSAVI3 FIXED(31) TO TASK SAVE AREA CIDSAVI3 FIXED(31) ADDR OF INIT ROUT CCHORD FIXED(31) ADDR OF UNIT ROUT CCHORD FIXED(31) ADDR TO CURR BINBLK CCURBAL FIXED(31) ADDR TO CURR BINBLK CCURBAL FIXED(31) ADDR TO CURR BINBLK CCUCRBL FIXED(31) ADDR TO CURR BINBLK CCURBAL FIXED(31) ADDR TO CURR BINBLK CCURBAL FIXED(31) ADDR OF OSTIN-OUT DECB 1EC CDATLOR FIXED(31) ADDR OF OSTIN-OUT DECB 1EC CDECCTR FIXED(31) ADDR OF OSTIN-OUT DECB 1EC CDECCTR FIXED(31) ADDR OF OSTIN-OUT DECB 1EC CDECCTR FIXED(31) ADDR OF OSTIN-OUT DECB 1EC CDEXCTR FIXED(31) ADDR OF OSTIN-OUT DECB 1EC CDESUG CHAR(1) DEBUG SMITCH BYTE CCEMPBUDE CHAR(1) DEBUG SMITCH BYTE CCEMPBUDE	Č	2_		PTR(31)	ADDR. OF TREEADDS ROUTINE
2 CBINBLK PTR(31) ADDR TO BINBLK Q 2 CBLKITTR PTR(31) NEXT UNUSED ITTR 2 CBLKITTR PTR(31) NEXT UNUSED ITTR 3 PTR(31) SIZE OF FREE BLOCK LIST 4 CBUFFER PTR(31) SIZE OF FREE BLOCK LIST 5 CBUFFTR PTR(31) IN PTR IN BUFF TO VLR 5 CBUFFTR PTR(31) IN PTR IN BUFF TO VLR 6 CCHIDX1 FIXED(31) INDEX AREA 1 HEAD 7 CCHIDX2 FIXED(31) INDEX AREA 2 HEAD 7 CCHIDX3 FIXED(31) POOL AREA 2 HEAD 8 CCHIDX4 FIXED(31) POOL AREA 1 HEAD 9 CCHIDX5 FIXED(31) POOL AREA 2 HEAD 9 CCHIDX5 FIXED(31) CONSTANT = 256 9 CCONNEG4 FIXED(31) CONSTANT = 256 1 CCONNEG4 FIXED(31) CONSTANT = 256 1 CCORESAV FIXED(31) CONSTANT = 256 1 CCORRESAV FIXED(31) CONSTANT = 256 2 CCORESAV FIXED(31) CONSTANT = 256 2 CCORRESAV FIXED(31) ADDR OF INIT ROUT 3 CIOSAVI3 FIXED(31) ADDR OF INIT ROUT 4 CTFCB FIXED(31) ADDR OF ORTHIN HEAD 2 CCURRBLK PTR(31) ADDR TO CURR BINBLK PROBLEM FIXED(31) ADDR TO FIREDEER BR-TABLE PROBLEM FIXED(31) ADDR OF TREEDER BR-TABLE PROBLEM F	•	_3		FIXED(31)	TREEGEN WORK AREA
2 CCHIDX1 FIXED(31) INDEX AREA 1 HEAD 2 CCHIDX3 FIXED(31) INDEX AREA 2 HEAD 3 CCHIDX3 FIXED(31) INDEX AREA 2 HEAD 4 CCHIDX5 FIXED(31) POOL AREA 2 HEAD 5 CCHIDX5 FIXED(31) POOL AREA 2 HEAD 6 CCHIDX5 FIXED(31) POOL AREA 2 HEAD 7 CCHIDX5 FIXED(31) CONSTANT = 256 8 CCON256 FIXED(31) CONSTANT = 256 8 CCON256 FIXED(31) CONSTANT = 256 8 CCONSEG FIXED(31) CONSTANT = 4 8 CCORESAV FIXED(31) ORIGINAL SIZE PARM 8 CCORESAV FIXED(31) ORIGINAL SIZE PARM 8 CCORESAV FIXED(31) ORIGINAL SIZE PARM 9 CCORESAV FIXED(31) ADDR OF INIT ROUT 1 CONSTANT = 4 9 COLONAL FIXED(31) ADDR OF INIT ROUT 1 CONSTANT = 4 1 COLONAL FIXED(31) ADDR OF INIT ROUT 1 COLONAL FIXED(31) ADDR OF INIT ROUT 2 CCURRBLK PIR(31) ADDR TO CURR BINBLK ADDR OF OPEN/CLOSE LIST 1 COLONAL FIXED(31) ADDR TO CURR BINBLK ADDR OF OPEN/CLOSE LIST 1 COLONAL FIXED(31) ADDR TO CURR BINBLK ADDR OF OPEN/CLOSE LIST 1 COLONAL FIXED(31) ADDR TO FORTIN/OUT DECB 1EC 2 COLONAL FIXED(31) ADDR OF OPEN/CLOSE LIST 2 COLONAL FIXED(31) ADDR OF OPEN/CLOSE LIST 2 COLONAL FIXED(31) ADDR OF TREELET ROUTINE ADDR OF TREELET ADDR OF T	1	2		FIXED(15)	BASE-BIN MIN ALLOC
2 CCHIDX1 FIXED(31) INDEX AREA 1 HEAD 2 CCHIDX3 FIXED(31) INDEX AREA 2 HEAD 3 CCHIDX3 FIXED(31) INDEX AREA 2 HEAD 4 CCHIDX5 FIXED(31) POOL AREA 2 HEAD 5 CCHIDX5 FIXED(31) POOL AREA 2 HEAD 6 CCHIDX5 FIXED(31) POOL AREA 2 HEAD 7 CCHIDX5 FIXED(31) CONSTANT = 256 8 CCON256 FIXED(31) CONSTANT = 256 8 CCON256 FIXED(31) CONSTANT = 256 8 CCONSEG FIXED(31) CONSTANT = 4 8 CCORESAV FIXED(31) ORIGINAL SIZE PARM 8 CCORESAV FIXED(31) ORIGINAL SIZE PARM 8 CCORESAV FIXED(31) ORIGINAL SIZE PARM 9 CCORESAV FIXED(31) ADDR OF INIT ROUT 1 CONSTANT = 4 9 COLONAL FIXED(31) ADDR OF INIT ROUT 1 CONSTANT = 4 1 COLONAL FIXED(31) ADDR OF INIT ROUT 1 COLONAL FIXED(31) ADDR OF INIT ROUT 2 CCURRBLK PIR(31) ADDR TO CURR BINBLK ADDR OF OPEN/CLOSE LIST 1 COLONAL FIXED(31) ADDR TO CURR BINBLK ADDR OF OPEN/CLOSE LIST 1 COLONAL FIXED(31) ADDR TO CURR BINBLK ADDR OF OPEN/CLOSE LIST 1 COLONAL FIXED(31) ADDR TO FORTIN/OUT DECB 1EC 2 COLONAL FIXED(31) ADDR OF OPEN/CLOSE LIST 2 COLONAL FIXED(31) ADDR OF OPEN/CLOSE LIST 2 COLONAL FIXED(31) ADDR OF TREELET ROUTINE ADDR OF TREELET ADDR OF T	ŧ	2		PIR(SI)	ADDK IN BINBLK O
2 CCHIDX1 FIXED(31) INDEX AREA 1 HEAD 2 CCHIDX3 FIXED(31) INDEX AREA 2 HEAD 3 CCHIDX3 FIXED(31) INDEX AREA 2 HEAD 4 CCHIDX4 FIXED(31) POOL AREA 2 HEAD 5 CCHIDX5 FIXED(31) POOL AREA 2 HEAD 6 CCHIDX5 FIXED(31) POOL AREA 2 HEAD 7 CCHIDX5 FIXED(31) CONSTANT = -256 8 CCON256 FIXED(31) CONSTANT = -256 9 CCON256 FIXED(31) CONSTANT = -256 9 CCON256 FIXED(31) CONSTANT = -4 10 CCONSTANT = -4 11 CCONSTANT = -4 12 CCORESAV FIXED(31) ORIGINAL SIZE PARM 12 W BDY(WORD) I/O INTERFACE FIELDS 13 CIOHOLD FIXED(31) ADDR OF INIT ROUT 14 CIOSAV13 FIXED(31) ADDR OF INIT ROUT 15 CIOSAV13 FIXED(31) I/O TASK SAVE AREA 14 CIOSAV13 FIXED(31) ADDR TO CURR BINBLK 16 CIOSAV13 FIXED(31) ADDR TO CURR BINBLK 17 CCURRBLK PIR(31) ADDR TO CURR BINBLK 18 CCURRBLK PIR(31) ADDR TO CURR BINBLK 19 CCURRBLK PIR(31) ADDR TO FORTIN/OUT DECB 1EC 2 CDATLOC BDY(WORD) ICEMAN BASEREGS 2 CDATLOC BDY(WORD) ICEMAN BASEREGS 2 CDELCNT FIXED(31) ADDR OF OF ORTIN/OUT DECB 1EC 2 CDECAREA PIR(31) ADDR OF FORTIN/OUT DECB 1EC 2 CDECAREA PIR(31) ADDR OF TREELETE ROUTINE 3 CREATE PIR(31) ADDR OF TREELETE ROUTINE 2 CDEXCTR FIXED(31) HI/LO INDEX COUNTER 2 CDEXCTR FIXED(31) HI/LO INDEX COUNTER 2 CDEXPTRK FIXED(31) INDEX ENTRIES PER TRACK 2 CDEXPTRK FIXED(31) INDEX ENTRIES PER TRACK 2 CDYNAMIC PIR(31) ADDR OF UNUSED DYNAMIC AREA 3 CITTRA FIXED(31) INDEX ENTRIES PER TRACK 4 COUNTING FIXED(31) ALTERNATE ITTR PTR 4 COUNTING FIXED(31) ALTERNATE ITTR PTR 5 COUNTING FIXED(31) ALTERNATE ITTR PTR 6 COUNTING FIXED(31) ALTERNATE ITTR PTR 7 COUNTING FIXED(31) ALTERNATE ITTR PTR 8 COUNTING FIXED(31) ALTERNATE ITTR PTR 9 COUNTING FIXED(31) ALTERNATE ITTR PTR 10 INTEX FOR ITTREE PER TRACK 10 COUNTING FIXED(31) ALTERNATE ITTR PTR 10 INTEX FOR ITTREE PER TRACK 11 COUNTING FIXED(31) ALTERNATE ITTR PTR 12 COUNTING FIXED(31) ALTERNATE ITTR PTR 13 CITTRA FIXED(31) ALTERNATE ITTR PTR 14 COUNTING FIXED(31) ALTERNATE ITTR PTR 15 COUNTING FIXED(31) ALTERNATE ITTR PTR 16 COUNTING FIXED(31) ALTERNATE OUTTING CEXTRACT POUTTING CEXTRACT POUTING CEXTRACT POUTTING CEXTRACT POUTTING CEXTRACT POUTTING CEXTRA	3	2		PIRCOLI	
2 CCHIDX1 FIXED(31) INDEX AREA 1 HEAD 2 CCHIDX3 FIXED(31) INDEX AREA 2 HEAD 3 CCHIDX3 FIXED(31) INDEX AREA 2 HEAD 4 CCHIDX4 FIXED(31) POOL AREA 2 HEAD 5 CCHIDX5 FIXED(31) POOL AREA 2 HEAD 6 CCHIDX5 FIXED(31) POOL AREA 2 HEAD 7 CCHIDX5 FIXED(31) CONSTANT = -256 8 CCON256 FIXED(31) CONSTANT = -256 9 CCON256 FIXED(31) CONSTANT = -256 9 CCON256 FIXED(31) CONSTANT = -4 10 CCONSTANT = -4 11 CCONSTANT = -4 12 CCORESAV FIXED(31) ORIGINAL SIZE PARM 12 W BDY(WORD) I/O INTERFACE FIELDS 13 CIOHOLD FIXED(31) ADDR OF INIT ROUT 14 CIOSAV13 FIXED(31) ADDR OF INIT ROUT 15 CIOSAV13 FIXED(31) I/O TASK SAVE AREA 14 CIOSAV13 FIXED(31) ADDR TO CURR BINBLK 16 CIOSAV13 FIXED(31) ADDR TO CURR BINBLK 17 CCURRBLK PIR(31) ADDR TO CURR BINBLK 18 CCURRBLK PIR(31) ADDR TO CURR BINBLK 19 CCURRBLK PIR(31) ADDR TO FORTIN/OUT DECB 1EC 2 CDATLOC BDY(WORD) ICEMAN BASEREGS 2 CDATLOC BDY(WORD) ICEMAN BASEREGS 2 CDELCNT FIXED(31) ADDR OF OF ORTIN/OUT DECB 1EC 2 CDECAREA PIR(31) ADDR OF FORTIN/OUT DECB 1EC 2 CDECAREA PIR(31) ADDR OF TREELETE ROUTINE 3 CREATE PIR(31) ADDR OF TREELETE ROUTINE 2 CDEXCTR FIXED(31) HI/LO INDEX COUNTER 2 CDEXCTR FIXED(31) HI/LO INDEX COUNTER 2 CDEXPTRK FIXED(31) INDEX ENTRIES PER TRACK 2 CDEXPTRK FIXED(31) INDEX ENTRIES PER TRACK 2 CDYNAMIC PIR(31) ADDR OF UNUSED DYNAMIC AREA 3 CITTRA FIXED(31) INDEX ENTRIES PER TRACK 4 COUNTING FIXED(31) ALTERNATE ITTR PTR 4 COUNTING FIXED(31) ALTERNATE ITTR PTR 5 COUNTING FIXED(31) ALTERNATE ITTR PTR 6 COUNTING FIXED(31) ALTERNATE ITTR PTR 7 COUNTING FIXED(31) ALTERNATE ITTR PTR 8 COUNTING FIXED(31) ALTERNATE ITTR PTR 9 COUNTING FIXED(31) ALTERNATE ITTR PTR 10 INTEX FOR ITTREE PER TRACK 10 COUNTING FIXED(31) ALTERNATE ITTR PTR 10 INTEX FOR ITTREE PER TRACK 11 COUNTING FIXED(31) ALTERNATE ITTR PTR 12 COUNTING FIXED(31) ALTERNATE ITTR PTR 13 CITTRA FIXED(31) ALTERNATE ITTR PTR 14 COUNTING FIXED(31) ALTERNATE ITTR PTR 15 COUNTING FIXED(31) ALTERNATE ITTR PTR 16 COUNTING FIXED(31) ALTERNATE OUTTING CEXTRACT POUTTING CEXTRACT POUTING CEXTRACT POUTTING CEXTRACT POUTTING CEXTRACT POUTTING CEXTRA		2		PTP(31)	
2 CCHIDX1 FIXED(31) INDEX AREA 1 HEAD 2 CCHIDX3 FIXED(31) INDEX AREA 2 HEAD 3 CCHIDX3 FIXED(31) INDEX AREA 2 HEAD 4 CCHIDX4 FIXED(31) POOL AREA 2 HEAD 5 CCHIDX5 FIXED(31) POOL AREA 2 HEAD 6 CCHIDX5 FIXED(31) POOL AREA 2 HEAD 7 CCHIDX5 FIXED(31) CONSTANT = 256 8 CCON256 FIXED(31) CONSTANT = 256 8 CCON256 FIXED(31) CONSTANT = 256 8 CCONSEG FIXED(31) ORIGINAL SIZE PARM 8 CCORESAV FIXED(31) ORIGINAL SIZE PARM 8 CCORESAV FIXED(31) ORIGINAL SIZE PARM 9 CCORESAV FIXED(31) ORIGINAL SIZE PARM 9 CCORESAV FIXED(31) ADDR OF INIT ROUT 1 CONSTANT = 4 9 COLONAL FIXED(31) ADDR OF INIT ROUT 1 COLONAL FIXED(31) ADDR OF INIT ROUT 1 COLONAL FIXED(31) ADDR OF INIT ROUT 2 CCURRBLK PIR(31) ADDR TO CURR BINBLK ADDR OF OPEN/CLOSE LIST 8 COLONAL FIXED(31) ADDR TO CURR BINBLK ADDR OF OPEN/CLOSE LIST 9 COLONAL FIXED(31) ADDR TO CURR BINBLK ADDR OF OPEN/CLOSE LIST 1 COLONAL FIXED(31) ADDR TO FORTIN/OUT DECB 1EC 1 COLONAL FIXED(31) ADDR OF OPEN/CLOSE LIST 1 COLONAL FIXED(31) ADDR OF OPEN/CLOSE LIST 2 COLONAL FIXED(31) ADDR OF TREELETE ROUTINE ADDR OF CONNAMIC AREA COLONAL PR(31) ADDR OF UNUSED DYNAMIC AREA COLONAL PR(31) ADDR OF UNUSED DYNAMIC AREA COLONAL PR(31) ADDR OF TREELETE ROUTINE ADDR OF CONNAMIC AREA COLONAL PR(31) ADDR OF CONNAMIC AREA COLONAL PROCESS COLONAL PROCESS COLONAL PROCESS COLONAL PROCESS COLONAL PROCESS COLONAL PROCESS COLON	}	2		FIXED(31)	SIZE OF EREE RIOCK LIST
2 CCHIDX1 FIXED(31) INDEX AREA 1 HEAD 2 CCHIDX3 FIXED(31) INDEX AREA 2 HEAD 3 CCHIDX3 FIXED(31) INDEX AREA 2 HEAD 4 CCHIDX4 FIXED(31) POOL AREA 2 HEAD 5 CCHIDX5 FIXED(31) POOL AREA 2 HEAD 6 CCHIDX5 FIXED(31) POOL AREA 2 HEAD 7 CCHIDX5 FIXED(31) CONSTANT = 256 8 CCON256 FIXED(31) CONSTANT = 256 8 CCON256 FIXED(31) CONSTANT = 256 8 CCONSEG FIXED(31) ORIGINAL SIZE PARM 8 CCORESAV FIXED(31) ORIGINAL SIZE PARM 8 CCORESAV FIXED(31) ORIGINAL SIZE PARM 9 CCORESAV FIXED(31) ORIGINAL SIZE PARM 9 CCORESAV FIXED(31) ADDR OF INIT ROUT 1 CONSTANT = 4 9 COLONAL FIXED(31) ADDR OF INIT ROUT 1 COLONAL FIXED(31) ADDR OF INIT ROUT 1 COLONAL FIXED(31) ADDR OF INIT ROUT 2 CCURRBLK PIR(31) ADDR TO CURR BINBLK ADDR OF OPEN/CLOSE LIST 8 COLONAL FIXED(31) ADDR TO CURR BINBLK ADDR OF OPEN/CLOSE LIST 9 COLONAL FIXED(31) ADDR TO CURR BINBLK ADDR OF OPEN/CLOSE LIST 1 COLONAL FIXED(31) ADDR TO FORTIN/OUT DECB 1EC 1 COLONAL FIXED(31) ADDR OF OPEN/CLOSE LIST 1 COLONAL FIXED(31) ADDR OF OPEN/CLOSE LIST 2 COLONAL FIXED(31) ADDR OF TREELETE ROUTINE ADDR OF CONNAMIC AREA COLONAL PR(31) ADDR OF UNUSED DYNAMIC AREA COLONAL PR(31) ADDR OF UNUSED DYNAMIC AREA COLONAL PR(31) ADDR OF TREELETE ROUTINE ADDR OF CONNAMIC AREA COLONAL PR(31) ADDR OF CONNAMIC AREA COLONAL PROCESS COLONAL PROCESS COLONAL PROCESS COLONAL PROCESS COLONAL PROCESS COLONAL PROCESS COLON		2	CBUFPTR	PTR(31)	IN - PTR IN BUFF TO VIR
CCHIDX1 FIXED(31) INDEX AREA 1 HEAD CCHIDX2 FIXED(31) INDEX AREA 2 HEAD CCHIDX3 FIXED(31) INDEX POOL HEAD CCHIDX4 FIXED(31) POOL AREA 1 HEAD CCHIDX5 FIXED(31) POOL AREA 1 HEAD CCHIDX5 FIXED(31) POOL AREA 2 HEAD CCONS56 FIXED(31) CONSTANT = 256 CCONNEG6 FIXED(31) CONSTANT = 256 CCONNEG6 FIXED(31) CONSTANT = -4 CCORESAV FIXED(31) LO INTERFACE FIELDS CCORESAV FIXED(31) LO INTERFACE FIELDS CCORESAV FIXED(31) ADDR OF INIT ROUT CCORESAV FIXED(31) ADDR OF OPEN/CLOSE LIST CCORESAV FIXED(31) ADDR OF CONTINIOUT DECB 1EC CCORESAV FIXED(31) ADDR OF SORTIN/OUT DECB 1EC CCORESAV FIXED(31) ADDR OF TREELETE ROUTINE CORENTAL FIXED(31) INDEX ENTRIES PER BUFFER CORENTAL FIXED(31) ALTERNATE ITTR FIXED CONNAMIC PIR(31) ALTERNATE ITTR FIXED COMPONE CHAR(1) DEBUG SWITCH BYTE COMPONE CHAR(1) ADDR OF EXTRACT ROUTINE CEXTRACT PIR(31) ADDR OF EXT		3		PTR(31)	
2 CCHIDX5 FIXED(31) POOL AREA 1 HEAD 2 CCHIDX5 FIXED(31) POOL AREA 2 HEAD 2 CCONSEG FIXED(31) CONSTANT = 256 2 CCONNEGG FIXED(31) CONSTANT = 256 2 CCONSEG FIXED(31) CONSTANT = 256 2 CCORESAV FIXED(31) ORIGINAL SIZE PARM 3 CIOHOLD FIXED(31) HOLD SW/Q END 4 CIOSAV13 FIXED(31) HOLD SW/Q END 5 CIOSAV13 FIXED(31) JODRO FINIT ROUT 5 CIOSAV13 FIXED(31) I/O TASK SAVE AREA 6 CTFCB FIXED(31) I/O TASK SAVE AREA 7 CTFCB FIXED(31) JODRO FINIT ROUT 8 COURRBLK PTR(31) ADDR OF OPEN/CLOSE LIST 9 COCURBLK PTR(31) ADDR OF OPEN/CLOSE LIST 1 COMBLIST PTR(31) ADDR OF OPEN/CLOSE LIST 1 COMBLES POR SOME SIME SIME SIME SIME SIME SIME SIME SI	;	2	CCHIDX1	FIXED(31)	INDEX AREA 1 HEAD
2 CCHIDX5 FIXED(31) POOL AREA 1 HEAD 2 CCHIDX5 FIXED(31) POOL AREA 2 HEAD 3 CCONESAV FIXED(31) CONSTANT = 256 2 CCONESAV FIXED(31) CONSTANT = 256 2 CCORESAV FIXED(31) ORIGINAL SIZE PARM 4 CIOHOLD FIXED(31) HOLD SW/Q END 4 CIOSAV13 FIXED(31) ADDR OF INIT ROUT 5 CIOSAV13 FIXED(31) I/O TASK SAVE AREA 4 CTFCB FIXED(31) I/O TASK SAVE AREA 4 CTFCB FIXED(31) ADDR OF INIT ROUT 5 COCURRBLK PIR(31) ADDR OF OPEN/CLOSE LIST 6 CDCBLIST PIR(31) ADDR OF OPEN/CLOSE LIST 7 CDCBLIST PIR(31) ADDR OF OPEN/CLOSE LIST 8 CDATLOC BDY(WORD) ICEMAN BASEREGS 9 CDATLOC BDY(WORD) ICEMAN BASEREGS 1 CDECAREA PIR(31) ADDR OF SORTIN/OUT DECB 1EC 1 CDELCRT FIXED(31) BYTES DELETE IN RSA 2 CDELCRT FIXED(31) ADDR OF TREELETE ROUTINE 3 CBRANCHP PIR(31) ADDR OF TREELETE ROUTINE 4 CDEXCTR FIXED(31) HI/LO INDEX COUNTER 5 CDEXCTR FIXED(31) HI/LO INDEX COUNTER 6 CDEXCTR FIXED(31) INDEX BHORE SHOFFER 7 CDEXCTR FIXED(31) INDEX ENTRIES PER BUFFER 7 CDEXCTR FIXED(31) INDEX ENTRIES PER BUFFER 8 CDEXCTR FIXED(31) INDEX ENTRIES PER BUFFER 9 CDEXCTR FIXED(31) INDEX ENTRIES PER BUFFER 9 CDEXCTR FIXED(31) INDEX ENTRIES PER TRACK 1 CDEXCTR FIXED(31) INDEX ENTRIES PER BUFFER 2 CDEXCTR FIXED(31) INDEX ENTRIES PER BUFFER 3 CDYNWK CHAR(8) DYN ALLOC DEVICE TYPE 4 CDYNWK CHAR(8) DYN ALLOC DEVICE TYPE 5 CDYNWK CHAR(8) DYN ALLOC DEVICE TYPE 6 CDYNWK CHAR(8) DYN ALLOC DEVICE TYPE 7 CDYNWK CHAR(8) DYN ALLOC DEVICE TYPE 8 CDYNWK CHAR(8) DYNALLOC FIELD FOR IMS 9 COUNTS TO THE STRING TO COUNTS TO THE STRING 1 CEXTRACT PIR(31) ADDR OF EXTRACT ROUTINE 1 CEXTRACT PIR(31) ADDR OF EXTRACT ROUTINE 2 CENTRACT PIR(31) ADDR OF EXTRACT ROUTINE 2 CEXTRACT PIR(31) ADDR OF EXTRACT ROUTINE	)	2		FIXED(31)	INDEX AREA 2 HEAD
2 X GIOHOLD FIXED(31) HOLD SW/Q END 4 CIOINIT FIXED(31) ADDR OF INIT ROUT 3 CIOSAV13 FIXED(31) I/O TASK SAVE AREA 4 CTFCB FIXED(31) TFCB CHAIN HEAD 2 CCURRBLK PTR(31) ADDR. OF OPEN/CLOSE LIST 2 CDCBLIST PTR(31) ADDR. OF OPEN/CLOSE LIST 2 CDATLOC BDY(WORD) ICEMAN BASEREGS 2 CDATLOC BDY(WORD) ICEMAN BASEREGS 2 CDECAREA PTR(31) ADDR. OF SORTIN/OUT DECB 1EC 3 CDECAREA PTR(31) ADDR. OF SORTIN/OUT DECB 1EC 4 CDELCNT FIXED(31) BYTES DELETE IN RSA 5 CDELETE PTR(31) ADDR. OF TREELETE ROUTINE 6 CDEXCTR FIXED(31) HI/LO INDEX COUNTER 7 CDEXCTR FIXED(31) HI/LO INDEX COUNTER 7 CDEXPTR PTR(31) ADDR. OF TREELETE ROUTINE 7 CDEXPTR FIXED(31) HI/LO INDEX ENTRIES PER BUFFER 7 CDEXPTR FIXED(31) INDEX ENTRIES PER BUFFER 7 CDEXPTR FIXED(31) INDEX ENTRIES PER TRACK 8 CDYNAMIC PTR(31) ADDR OF UNUSED DYNAMIC AREA 9 COYNNUM FIXED(31) ALTERNATE ITTR PTR 9 CHAR(3) REST OF CDYNIWK 1ST WORD 9 ALLECNATE ITTR PTR 1 CDYNUMB FIXED(31) ALTERNATE ITTR PTR 1 CDYNAMIC PTR(8) DYNALLOC DEVICES 1 COTTRE FIXED(31) ALTERNATE ITTR PTR 1 CDYNAMIC PTR(8) DYNALLOC FIELD LENGTH 1 CDYNALD FIXED(31) AUBBER OF E-PHASE BUFFERS 1 CDYNAMIC PTR(8) DYNALLOC FIELD LENGTH 1 CDYNALD FIXED(31) ADDR. OF EXTRACT ROUTINE 1 CDYNAMIC PTR(31) DEBUG SWITCH BYTE 1 CDYNAMIC PTR(31) ADDR. OF EXTRACT ROUTINE 2 CEXPAND FIXED(31) INT LRECL EXPAND POS OR NEG 2 CEXPAND FIXED(31) INT LRECL EXPAND POS OR NEG 2 CEXPAND FIXED(31) MERGE ORDER - ELIMINATION	4	2		FIXED(31)	INDEX POOL HEAD
2 X GIOHOLD FIXED(31) HOLD SW/Q END 4 CIOINIT FIXED(31) ADDR OF INIT ROUT 3 CIOSAV13 FIXED(31) I/O TASK SAVE AREA 4 CTFCB FIXED(31) TFCB CHAIN HEAD 2 CCURRBLK PTR(31) ADDR. OF OPEN/CLOSE LIST 2 CDCBLIST PTR(31) ADDR. OF OPEN/CLOSE LIST 2 CDATLOC BDY(WORD) ICEMAN BASEREGS 2 CDATLOC BDY(WORD) ICEMAN BASEREGS 2 CDECAREA PTR(31) ADDR. OF SORTIN/OUT DECB 1EC 3 CDECAREA PTR(31) ADDR. OF SORTIN/OUT DECB 1EC 4 CDELCNT FIXED(31) BYTES DELETE IN RSA 5 CDELETE PTR(31) ADDR. OF TREELETE ROUTINE 6 CDEXCTR FIXED(31) HI/LO INDEX COUNTER 7 CDEXCTR FIXED(31) HI/LO INDEX COUNTER 7 CDEXPTR PTR(31) ADDR. OF TREELETE ROUTINE 7 CDEXPTR FIXED(31) HI/LO INDEX ENTRIES PER BUFFER 7 CDEXPTR FIXED(31) INDEX ENTRIES PER BUFFER 7 CDEXPTR FIXED(31) INDEX ENTRIES PER TRACK 8 CDYNAMIC PTR(31) ADDR OF UNUSED DYNAMIC AREA 9 COYNNUM FIXED(31) ALTERNATE ITTR PTR 9 CHAR(3) REST OF CDYNIWK 1ST WORD 9 ALLECNATE ITTR PTR 1 CDYNUMB FIXED(31) ALTERNATE ITTR PTR 1 CDYNAMIC PTR(8) DYNALLOC DEVICES 1 COTTRE FIXED(31) ALTERNATE ITTR PTR 1 CDYNAMIC PTR(8) DYNALLOC FIELD LENGTH 1 CDYNALD FIXED(31) AUBBER OF E-PHASE BUFFERS 1 CDYNAMIC PTR(8) DYNALLOC FIELD LENGTH 1 CDYNALD FIXED(31) ADDR. OF EXTRACT ROUTINE 1 CDYNAMIC PTR(31) DEBUG SWITCH BYTE 1 CDYNAMIC PTR(31) ADDR. OF EXTRACT ROUTINE 2 CEXPAND FIXED(31) INT LRECL EXPAND POS OR NEG 2 CEXPAND FIXED(31) INT LRECL EXPAND POS OR NEG 2 CEXPAND FIXED(31) MERGE ORDER - ELIMINATION	8	2		FIXED(31)	POOL AREA 1 HEAD
2 X GIOHOLD FIXED(31) HOLD SW/Q END 4 CIOINIT FIXED(31) ADDR OF INIT ROUT 3 CIOSAVI3 FIXED(31) I/O TASK SAVE AREA 4 CTFCB FIXED(31) TFCB CHAIN HEAD 2 CCURRBLK PTR(31) ADDR. OF OPEN/CLOSE LIST 2 CDCBLIST PTR(31) ADDR. OF OPEN/CLOSE LIST 2 CDATLOC BDY(WORD) ICEMAN BASEREGS 2 CDATLOC BDY(WORD) ICEMAN BASEREGS 2 CDECAREA PTR(31) ADDR. OF SORTIN/OUT DECB 1EC 3 CDECAREA PTR(31) ADDR. OF SORTIN/OUT DECB 1EC 4 CDELCNT FIXED(31) BYTES DELETE IN RSA 5 CDELETE PTR(31) ADDR. OF TREELETE ROUTINE 6 CDEXCTR FIXED(31) HI/LO INDEX COUNTER 7 CDEXCTR FIXED(31) HI/LO INDEX COUNTER 7 CDEXPTR FIXED(31) INDEX ENTRIES PER BUFFER 7 CDEXPTR FIXED(31) INDEX ENTRIES PER BUFFER 7 CDEXPTR FIXED(31) INDEX ENTRIES PER RACK 7 CDEXPTR FIXED(31) INDEX ENTRIES PER TRACK 8 CDYNAMIC PTR(31) ADDR OF UNUSED DYNAMIC AREA 9 COYNNUMB FIXED(31) ALTERNATE ITTR PTR 9 COYNNUMB FIXED(31) ALTERNATE ITTR PTR 1 CDYNAMIC PTR(8) DYNALLOC DEVICES 1 CTITR FIXED(31) ALTERNATE ITTR PTR 1 CDYNAMIC PTR(8) DYNALLOC FIELD LENGTH 1 CDYNALD FIXED(31) AUBBER OF E-PHASE BUFFERS 1 CDYNAMIC PTR(8) DYNALLOC FIELD LENGTH 1 CDYNALD FIXED(31) ADDR. OF EXTRACT ROUTINE 1 CDYNAMIC PTR(31) DEBUG SWITCH BYTE 1 CDEBUG CHAR(1) DEBUG SWITCH BYTE 2 CENDMODE CHAR(1) NO = UNEXPECTED SYSIN EOF 2 CEXTRACT PTR(31) ADDR. OF EXTRACT ROUTINE 2 CEXPAND FIXED(31) INT LRECL EXPAND POS OR NEG 2 CEXPAND FIXED(31) INT LRECL EXPAND POS OR NEG 2 CEMERGE FIXED(31) MERGE ORDER - ELIMINATION	Č	2	CCHIDX5	FIXED(31)	
2 X GIOHOLD FIXED(31) HOLD SW/Q END 4 CIOINIT FIXED(31) ADDR OF INIT ROUT 3 CIOSAVI3 FIXED(31) I/O TASK SAVE AREA 4 CTFCB FIXED(31) TFCB CHAIN HEAD 2 CCURRBLK PTR(31) ADDR. OF OPEN/CLOSE LIST 2 CDCBLIST PTR(31) ADDR. OF OPEN/CLOSE LIST 2 CDATLOC BDY(WORD) ICEMAN BASEREGS 2 CDATLOC BDY(WORD) ICEMAN BASEREGS 2 CDECAREA PTR(31) ADDR. OF SORTIN/OUT DECB 1EC 2 CDECAREA PTR(31) ADDR. OF SORTIN/OUT DECB 1EC 2 CDELCNT FIXED(31) BYTES DELETE IN RSA 2 CDELETE PTR(31) ADDR. OF TREELETE ROUTINE 3 CBRANCHP PTR(31) ADDR. OF TREELETE ROUTINE 4 CDEXCTR FIXED(31) HI/LO INDEX COUNTER 5 CDEXCTR FIXED(31) HI/LO INDEX COUNTER 5 CDEXPBUF FIXED(31) INDEX ENTRIES PER BUFFER 6 CDEXPTRK FIXED(31) INDEX ENTRIES PER BUFFER 7 CDEXPTRK FIXED(31) INDEX ENTRIES PER TRACK 8 CDYNAMIC PTR(31) ADDR OF UNUSED DYNAMIC AREA 9 COYNNUMB FIXED(31) INTERNATE ITTR PTR 1 CHAR(3) REST OF COYNNUK 1ST WORD 1 ST BYTE OF DYN DEV TYPE 1 CDYNNUMB FIXED(31) ALTERNATE ITTR PTR 1 CDYNUMB FIXED(31) ALTERNATE ITTR PTR 2 CDYNNUMB FIXED(31) ALTERNATE ITTR PTR 2 CDYNLEN PTR(8) DYNALLOC DEVICES 1 CTITR FIXED(31) ALTERNATE ITTR PTR 2 CDYNLEN PTR(8) DYNALLOC FIELD LENGTH 1 CDYNCE TYPE OF DYN ALLOC DEVICES 2 CDEBUG CHAR(1) DEBUG SWITCH BYTE 2 CDEBUG CHAR(1) DEBUG SWITCH BYTE 2 CEXTRACT PTR(31) ADDR. OF EXTRACT ROUTINE 2 CEXPAND FIXED(31) INT LRECL EXPAND POS OR NEG 2 CEXPAND FIXED(31) INT LRECL EXPAND POS OR NEG	0	2		FIXED(31)	
2 X GIOHOLD FIXED(31) HOLD SW/Q END 4 CIOINIT FIXED(31) ADDR OF INIT ROUT 3 CIOSAVI3 FIXED(31) I/O TASK SAVE AREA 4 CTFCB FIXED(31) TFCB CHAIN HEAD 2 CCURRBLK PTR(31) ADDR. OF OPEN/CLOSE LIST 2 CDCBLIST PTR(31) ADDR. OF OPEN/CLOSE LIST 2 CDATLOC BDY(WORD) ICEMAN BASEREGS 2 CDATLOC BDY(WORD) ICEMAN BASEREGS 2 CDECAREA PTR(31) ADDR. OF SORTIN/OUT DECB 1EC 2 CDECAREA PTR(31) ADDR. OF SORTIN/OUT DECB 1EC 2 CDELCNT FIXED(31) BYTES DELETE IN RSA 2 CDELETE PTR(31) ADDR. OF TREELETE ROUTINE 3 CBRANCHP PTR(31) ADDR. OF TREELETE ROUTINE 4 CDEXCTR FIXED(31) HI/LO INDEX COUNTER 5 CDEXCTR FIXED(31) HI/LO INDEX COUNTER 5 CDEXPBUF FIXED(31) INDEX ENTRIES PER BUFFER 6 CDEXPTRK FIXED(31) INDEX ENTRIES PER BUFFER 7 CDEXPTRK FIXED(31) INDEX ENTRIES PER TRACK 8 CDYNAMIC PTR(31) ADDR OF UNUSED DYNAMIC AREA 9 COYNNUMB FIXED(31) INTERNATE ITTR PTR 1 CHAR(3) REST OF COYNNUK 1ST WORD 1 ST BYTE OF DYN DEV TYPE 1 CDYNNUMB FIXED(31) ALTERNATE ITTR PTR 1 CDYNUMB FIXED(31) ALTERNATE ITTR PTR 2 CDYNNUMB FIXED(31) ALTERNATE ITTR PTR 2 CDYNLEN PTR(8) DYNALLOC DEVICES 1 CTITR FIXED(31) ALTERNATE ITTR PTR 2 CDYNLEN PTR(8) DYNALLOC FIELD LENGTH 1 CDYNCE TYPE OF DYN ALLOC DEVICES 2 CDEBUG CHAR(1) DEBUG SWITCH BYTE 2 CDEBUG CHAR(1) DEBUG SWITCH BYTE 2 CEXTRACT PTR(31) ADDR. OF EXTRACT ROUTINE 2 CEXPAND FIXED(31) INT LRECL EXPAND POS OR NEG 2 CEXPAND FIXED(31) INT LRECL EXPAND POS OR NEG	:4 :8	2		LIXED(31)	
CIUNNI FIXED(31) ADDR OF INIT ROUT  CIUSAVI3 FIXED(31) I/O TASK SAVE AREA  CTFCB FIXED(31) TFCB CHAIN HEAD  CCURRBLK PTR(31) ADDR TO CURR BINBLK  CCURRBLK PTR(31) ADDR. OF OPEN/CLOSE LIST  CDATLOC BDY(WORD) ICEMAN BASEREGS  CDATLEN FIXED(31) DATA BLOCK SIZE  CDECAREA PTR(31) ADDR. OF SORTIN/OUT DECB IEC  CDELCNT FIXED(31) BYTES DELETE IN RSA  CDELCNT FIXED(31) ADDR. OF TREELETE ROUTINE  COBERANCHP PTR(31) ADDR. OF TREELETE ROUTINE  CDEXCTR FIXED(31) HI/LO INDEX COUNTER  CDEXCTR FIXED(31) HI/LO INDEX COUNTER  CDEXCTR FIXED(31) HI/LO INDEX COUNTER  CDEXPTRK FIXED(31) INDEX ENTRIES PER BUFFER  CDEXPTRK FIXED(31) INDEX ENTRIES PER TRACK  CDYNAMIC PTR(31) ADDR OF UNUSED DYNAMIC AREA  CDYNWK CHAR(8) DYN ALLOC DEVICE TYPE  COYNNUK PTR(8) IST BYTE OF DYN DEV TYPE  CHAR(3) REST OF CDYNWK IST WORD  CITTRA FIXED(31) ALTERNATE ITTR PTR  CHAR(3) REST OF CDYNWK IST WORD  COYNLEN PTR(8) DYNALLOC DEVICES  COYNLEN PTR(8) DYNALLOC FIELD LENGTH  COMMIC CHAR(1) NUMBER OF E-PHASE BUFFERS  COMMIC CHAR(1) NO = UNEXPECTED SYSIN EOF  CEBUFFRS FIXED(16) NUMBER OF E-PHASE BUFFERS  CDEBUG CHAR(1) NO = UNEXPECTED SYSIN EOF  CEXTRACT PTR(31) ADDR. OF EXTRACT ROUTINE  CEXTRACT PTR(31) INT LRECL EXPAND POS OR NEG  CEMERGE FIXED(31) INT LRECL EXPAND POS OR NEG	•	2		LIYER(21)	
CIUNNI FIXED(31) ADDR OF INIT ROUT  CIOSAV13 FIXED(31) I/O TASK SAVE AREA  CTFCB FIXED(31) TFCB CHAIN HEAD  CCURRBLK PTR(31) ADDR TO CURR BINBLK  CCURRBLK PTR(31) ADDR. OF OPEN/CLOSE LIST  CDEBLIST PTR(31) ADDR. OF OPEN/CLOSE LIST  CDATLOC BDY(WORD) ICEMAN BASEREGS  CDATLEN FIXED(31) DATA BLOCK SIZE  CDECAREA PTR(31) ADDR. OF SORTIN/OUT DECB 1EC  CDELCNT FIXED(31) BYTES DELETE IN RSA  CDELCNT FIXED(31) ADDR. OF TREELETE ROUTINE  COBERANCHP PTR(31) ADDR. OF TREELETE ROUTINE  CDEXCTR FIXED(31) HI/LO INDEX COUNTER  CDEXCTR FIXED(31) HI/LO INDEX COUNTER  CDEXCTR FIXED(31) HI/LO INDEX COUNTER  CDEXPTRK FIXED(31) INDEX ENTRIES PER BUFFER  CDEXPTRK FIXED(31) INDEX ENTRIES PER BUFFER  CDEXPTRK FIXED(31) INDEX ENTRIES PER BUFFER  CDYNWK CHAR(8) DYN ALLOC DEVICE TYPE  COMMIC PTR(31) ADDR OF UNUSED DYNAMIC AREA  CDYNNWK CHAR(8) DYN ALLOC DEVICE TYPE  CHAR(3) REST OF CDYNWK 1ST WORD  COYNNUMB FIXED(31) ALTERNATE ITTR PTR  CHAR(3) REST OF CDYNWK 1ST WORD  COTTRA FIXED(31) ALTERNATE ITTR PTR  CHAR(3) REST OF CDYNWK 1ST WORD  COYNLEN PTR(8) DYNALLOC FIELD LENGTH  COMMIC PTR(8) DYNALLOC PTR(8)  COMMIC PTR(8) DYNALLOC PT	C	ِ ع		EIYEN(31)	
GUELETE PTR(31) ADDR. OF FREEGEN BR-TABLE CDELETE PTR(31) ADDR. OF TREELETE ROUTINE COBENTY PTR(31) INDEX BLOCK ITTR COBENTY PTR(31) INDEX BLOCK ITTR COBENTY PTR(31) INDEX ENTRIES PER BUFFER CODYNAMIC PTR(31) INDEX ENTRIES PER BUFFER CODYNAMIC PTR(31) ADDR OF UNUSED DYNAMIC AREA CODYNAMIC PTR(31) ADDR OF CODYNAMIC AREA CODYNAMIC PTR(31) ALTERNATE ITTR PTR CHAR(3) REST OF CODYNAMIC STRING CODYNAMIC PTR(31) ALTERNATE ITTR PTR CHAR(3) REST OF CODYNAMIC STRING CODYNAMIC PTR(31) ALTERNATE ITTR PTR CODYNAMIC PTR(31) ALTERNATE PTR TO 1ST INP CODYNAMIC PTR(31) ALTERNATE PTR TO 1ST INP CODYNAMIC PTR(31) ALTERNATE PTR TO 1ST INP CO		<b>~</b> 4		FIXED(31)	
4 CTFCB FIXED(31) TFCB CHAIN HEAD CCURRBLK PIR(31) ADDR TO CURR BINBLK CDCBLIST PTR(31) ADDR TO CURR BINBLK CDCBLIST PTR(31) ADDR TO CURR BINBLK CDATLOC BDY(WORD) ICEMAN BASEREGS CDATLEN FIXED(31) DATA BLOCK SIZE CDECAREA PIR(31) ADDR. OF SORTIN/OUT DECB 1EC CDELCNT FIXED(31) BYTES DELETE IN RSA CDELETE PIR(31) ADDR. OF TREELETE ROUTINE CDEXCTR FIXED(31) HI/LO INDEX COUNTER CDEXCTR FIXED(31) HI/LO INDEX COUNTER CDEXCTR FIXED(31) HI/DEX BLOCK ITTR CDEXCPBUF FIXED(31) INDEX ENTRIES PER BUFFER CDEXCPRK FIXED(31) INDEX ENTRIES PER BUFFER CDEXCPRK FIXED(31) INDEX ENTRIES PER TRACK CDEXCOUNTER	Ō			FIXED(31)	I/O TASK SAVE AREA
CCURRBLK PTR(31) ADDR TO CURR BINBLK CDCBLIST PTR(31) ADDR. OF OPEN/CLOSE LIST CDATLOC BDY(WORD) ICEMAN BASEREGS CDATLEN FIXED(31) DATA BLOCK SIZE CDECAREA PTR(31) ADDR. OF SORTIN/OUT DECB 1EC CDELCNT FIXED(31) BYTES DELETE IN RSA CDELETE PTR(31) ADDR. OF TREEBER BR-TABLE CDEXCTR FIXED(31) HI/LO INDEX COUNTER CDEXCTR FIXED(31) HI/LO INDEX COUNTER CDEXITTR PTR(31) INDEX BLOCK ITTR CDEXPTRK FIXED(31) INDEX ENTRIES PER BUFFER CDEXPTRK FIXED(31) INDEX ENTRIES PER TRACK CDYNAMIC PTR(31) ADDR OF UNUSED DYNAMIC AREA CDYNAMIC PTR(31) ALTERNATE ITTR PTR CHAR(3) REST OF CDYNWK 1ST WORD CLITTR FIXED(31) ALTERNATE ITTR PTR CHAR(3) REST OF CDYNWK 1ST WORD CLITTR FIXED(31) ALTERNATE ITTR PTR CCDYNLEN FIXED(31) ITTR PTR TO 1ST INP STRING CDYNLEN PTR(8) DYNALLOC DEVICES CDYNLEN PTR(8) DYNALLOC DEVICES CDYNLEN PTR(8) DYNALLOC FIELD LENGTH CDEBUG CHAR(1) NUMBER OF E-PHASE BUFFERS CDEBUG CHAR(1) NO = UNEXPECTED SYSIN EOF CEXTRACT PTR(31) ADDR. OF EXTRACT ROUTINE	Ō			FIXED(31)	TECB CHAIN HEAD
CDCBLIST PTR(31) ADDR. OF OPEN/CLOSE LIST CDATLOC BDY(WORD) ICEMAN BASEREGS CDATLEN FIXED(31) DATA BLOCK SIZE CDECAREA PTR(31) ADDR. OF SORTIN/OUT DECB 1EC CDELCNT FIXED(31) BYTES DELETE IN RSA CDELETE PTR(31) ADDR. OF TREELETE ROUTINE CBRANCHP PTR(31) ADDR. OF TREEGEN BR-TABLE CDEXCTR FIXED(31) HI/LO INDEX COUNTER CDEXITTR PTR(31) INDEX BLOCK ITTR CDEXPBUF FIXED(31) INDEX ENTRIES PER BUFFER CDEXPTRK FIXED(31) INDEX ENTRIES PER TRACK CDYNAMIC PTR(31) ADDR OF UNUSED DYNAMIC AREA CDYNAWK CHAR(8) DYN ALLOC DEVICE TYPE  CDYNAWK FIXED(31) ALTERNATE ITTR PTR CDYNHUMB FIXED(31) ALTERNATE ITTR PTR COYNLEN FIXED(31) INTR PTR TO 1ST INP STRING CDIMSHI FIXED(31) ITTR PTR TO 1ST INP STRING CDIMSHI FIXED(31) ITTR PTR TO 1ST INP STRING CDIMSHI FIXED(31) HI CONTROL FIELD FOR IMS CDYNLEN PTR(8) DYNALLOC FIELD LENGTH CDEBUG CHAR(1) DEBUG SWITCH BYTE CEBUFFRS FIXED(16) HUMBER OF E-PHASE BUFFERS CDEBUG CHAR(1) NO = UNEXPECTED SYSIN EOF CEXTRACT PTR(31) ADDR. OF EXTRACT ROUTINE	4	2		PTR(31)	
CDELCNT FIXED(31) BYTES DELETE IN RSA CDELETE PTR(31) ADDR. OF TREELETE ROUTINE ADDR. OF TREELETE ROUTINE CDEXCTR FIXED(31) HI/LO INDEX COUNTER CDEXCTR FIXED(31) HI/LO INDEX COUNTER CDEXTITR PTR(31) INDEX BLOCK ITTR CDEXPBUF FIXED(31) INDEX ENTRIES PER BUFFER CDEXPTRK FIXED(31) INDEX ENTRIES PER TRACK CDEXPTRK FIXED(31) INDEX ENTRIES PER TRACK CDYNAMIC PTR(31) ADDR OF UNUSED DYNAMIC AREA CDYNAWK CHAR(8) DYN ALLOC DEVICE TYPE COMMIC PTR(8) 1ST BYTE OF DYN DEV TYPE CHAR(3) REST OF CDYNWK 1ST WORD CITTRA FIXED(31) ALTERNATE ITTR PTR COMMINIMB FIXED(31) HUMB OF DYN ALLOC DEVICES COMMINIMB FIXED(31) ITTR PTR TO 1ST INP STRING CDIMSHI FIXED(31) HITT PTR TO 1ST INP STRING CDYNLEN PTR(8) DYNALLOC FIELD FOR IMS COMMINIMS FIXED(16) HI CONTROL FIELD FOR IMS COMMINIMS FIXED(16) NUMBER OF E-PHASE BUFFERS CDEBUG CHAR(1) DEBUG SWITCH BYTE CEBUFFRS FIXED(16) NUMBER OF E-PHASE BUFFERS CDEBUG CHAR(1) NO = UNEXPECTED SYSIN EOF CEXTRACT PTR(31) ADDR. OF EXTRACT ROUTINE CEMERGE FIXED(31) MERGE ORDER - ELIMINATION	8	2	CDCBLIST	PTR(31)	
CDELCNT FIXED(31) BYTES DELETE IN RSA CDELETE PTR(31) ADDR. OF TREELETE ROUTINE ADDR. OF TREELETE ROUTINE CDEXCTR FIXED(31) HI/LO INDEX COUNTER CDEXCTR FIXED(31) HI/LO INDEX COUNTER CDEXCTR FIXED(31) INDEX BLOCK ITTR CDEXCHITR PTR(31) INDEX ENTRIES PER BUFFER CDEXCHITK FIXED(31) INDEX ENTRIES PER TRACK CDEXCHITK FIXED(31) INDEX ENTRIES PER TRACK CDEXCHITK FIXED(31) ADDR OF UNUSED DYNAMIC AREA CDYNAMIC PTR(31) ADDR OF UNUSED DYNAMIC AREA CDYNAWK CHAR(8) DYN ALLOC DEVICE TYPE COLYNWK CHAR(8) DYN ALLOC DEVICE TYPE CHAR(3) REST OF CDYNWK 1ST WORD CITTRA FIXED(31) ALTERNATE ITTR PTR COLYNUMB FIXED(31) NUMB OF DYN ALLOC DEVICES CITTR FIXED(31) ITTR PTR TO 1ST INP STRING CDYNLEN FIXED(31) HI CONTROL FIELD FOR IMS CDYNLEN PTR(8) DYNALLOC FIELD LENGTH CEBUFFRS FIXED(16) NUMBER OF E-PHASE BUFFERS CDEBUG CHAR(1) DEBUG SWITCH BYTE CENDMODE CHAR(1) NO = UNEXPECTED SYSIN EOF CEXTRACT PTR(31) ADDR. OF EXTRACT ROUTINE CEMERGE FIXED(31) INT LRECL EXPAND POS OR NEG	Ç	2		BDY(WORD)	
CDELCNT FIXED(31) BYTES DELETE IN RSA CDELETE PTR(31) ADDR. OF TREELETE ROUTINE BYTES DELETE IN RSA CDERANCHP PTR(31) ADDR. OF TREELETE ROUTINE CDEXCTR FIXED(31) HI/LO INDEX COUNTER CDEXCTR FIXED(31) HI/LO INDEX COUNTER CDEXPBUF FIXED(31) INDEX BLOCK ITTR CDEXPBUF FIXED(31) INDEX ENTRIES PER BUFFER CDEXPTRK FIXED(31) INDEX ENTRIES PER TRACK CDYNAMIC PTR(31) ADDR OF UNUSED DYNAMIC AREA CDYNAWK CHAR(8) DYN ALLOC DEVICE TYPE CDYNWK CHAR(8) DYN ALLOC DEVICE TYPE CHAR(3) REST OF CDYNWK 1ST WORD CITTRA FIXED(31) ALTERNATE ITTR PTR CDYNNUMB FIXED(31) NUMB OF DYN ALLOC DEVICES CITTR FIXED(31) ITTR PTR TO 1ST INP STRING CDYNLEN FIXED(31) HI CONTROL FIELD FOR IMS CDYNLEN PTR(8) DYNALLOC FIELD LENGTH CDEBUG CHAR(1) DEBUG SWITCH BYTE CEBUFFRS FIXED(16) NUMBER OF E-PHASE BUFFERS CDEBUG CHAR(1) NO = UNEXPECTED SYSIN EOF CEXTRACT PTR(31) ADDR. OF EXTRACT ROUTINE CEXTRACT PTR(31) INT LRECL EXPAND POS OR NEG CEMERGE FIXED(31) MERGE ORDER - ELIMINATION	4	2	CDATLEN	FIXED(31)	DATA BLOCK SIZE
CDELETE PTR(31) ADDR. OF TREELETE ROUTINE CBRANCHP PTR(31) ADDR. OF TREEGEN BR-TABLE CDEXCTR FIXED(31) HI/LO INDEX COUNTER CDEXITTR PTR(31) INDEX BLOCK ITTR CDEXPBUF FIXED(31) INDEX ENTRIES PER BUFFER CDEXPTRK FIXED(31) INDEX ENTRIES PER TRACK CDYNAMIC PTR(31) ADDR OF UNUSED DYNAMIC AREA CDYNWK CHAR(8) DYN ALLOC DEVICE TYPE CDYNWK CHAR(8) 1ST BYTE OF DYN DEV TYPE CHAR(3) REST OF CDYNWK 1ST WORD CITTRA FIXED(31) ALTERNATE ITTR PTR CDYNNUMB FIXED(31) NUMB OF DYN ALLOC DEVICES CITTR FIXED(31) ITTR PTR TO 1ST INP STRING CDIMSHI FIXED(31) HI CONTROL FIELD FOR IMS CDYNLEN PTR(8) DYNALLOC FIELD LENGTH CDYNLEN PTR(8) DYNALLOC FIELD LENGTH CEBUFFRS FIXED(16) NUMBER OF E-PHASE BUFFERS CDEBUG CHAR(1) DEBUG SWITCH BYTE CENDMODE CHAR(1) NO = UNEXPECTED SYSIN EOF CEXTRACT PTR(31) ADDR. OF EXTRACT ROUTINE CEXTRACT PTR(31) INT LRECL EXPAND POS OR NEG CEMERGE FIXED(31) MERGE ORDER - ELIMINATION	8	2		PIR(31)	ADDR. OF SORTIN/OUT DECB 1EC
CBRANCHP PTR(31) ADDR. OF TREEGEN BR-TABLE CDEXCTR FIXED(31) HI/LO INDEX COUNTER CDEXITTR PTR(31) INDEX BLOCK ITTR CDEXPBUF FIXED(31) INDEX ENTRIES PER BUFFER CDEXPTRK FIXED(31) INDEX ENTRIES PER TRACK CDEXPTRK FIXED(31) ADDR OF UNUSED DYNAMIC AREA CDYNMK CHAR(8) DYN ALLOC DEVICE TYPE CDYNMK1 PTR(8) 1ST BYTE OF DYN DEV TYPE CHAR(3) REST OF CDYNMK 1ST WORD CITTRA FIXED(31) ALTERNATE ITTR PTR CDYNNUMB FIXED(31) NUMB OF DYN ALLOC DEVICES CITTR FIXED(31) ITTR PTR TO 1ST INP STRING CDIMSHI FIXED(31) ITTR PTR TO 1ST INP STRING CDIMSHI FIXED(16) HI CONTROL FIELD FOR IMS CDYNLEN PTR(8) DYNALLOC FIELD ENGTH CEBUFFRS FIXED(16) NUMBER OF E-PHASE BUFFERS CDEBUG CHAR(1) DEBUG SWITCH BYTE CENDMODE CHAR(1) NO = UNEXPECTED SYSIN EOF CEXTRACT PTR(31) ADDR. OF EXTRACT ROUTINE CEXTRACT PTR(31) INT LRECL EXPAND POS OR NEG CEMERGE FIXED(31) MERGE ORDER - ELIMINATION	0 4	2			
CDEXCTR FIXED(31) HI/LO INDEX COUNTER CDEXITTR PTR(31) INDEX BLOCK ITTR CDEXPBUF FIXED(31) INDEX ENTRIES PER BUFFER CDEXPTRK FIXED(31) INDEX ENTRIES PER TRACK CDEXPTRK FIXED(31) INDEX ENTRIES PER TRACK CDYNAMIC PTR(31) ADDR OF UNUSED DYNAMIC AREA CDYNWK CHAR(8) DYN ALLOC DEVICE TYPE CDYNWK CHAR(8) DYN ALLOC DEVICE TYPE CDYNNUK FIXED(31) ALTERNATE ITTR PTR CDYNNUMB FIXED(31) NUMB OF DYN ALLOC DEVICES COTITR FIXED(31) ITTR PTR TO 1ST INP STRING CDIMSHI FIXED(31) ITTR PTR TO 1ST INP STRING CDIMSHI FIXED(31) HI CONTROL FIELD FOR IMS CDIMSHI FIXED(16) HI CONTROL FIELD FOR IMS CDIMSHI FIXED(16) HI CONTROL FIELD FOR IMS CDEBUFFRS FIXED(16) NUMBER OF E-PHASE BUFFERS CDEBUG CHAR(1) DEBUG SWITCH BYTE CEBUFFRS FIXED(31) ADDR. OF EXTRACT ROUTINE CEXTRACT PTR(31) ADDR. OF EXTRACT ROUTINE CEXTRACT FIXED(31) INT LRECL EXPAND POS OR NEG CEMERGE FIXED(31) MERGE ORDER - ELIMINATION	•	<u> </u>		L   K(OT)	ADDR. OF TREELEIE KUUIINE
CDEXITTR PTR(31) INDEX BLOCK ITTR CDEXPBUF FIXED(31) INDEX ENTRIES PER BUFFER CDEXPTRK FIXED(31) INDEX ENTRIES PER TRACK CDYNAMIC PTR(31) ADDR OF UNUSED DYNAMIC AREA CDYNWK CHAR(8) DYN ALLOC DEVICE TYPE CDYNWK1 PTR(8) 1ST BYTE OF DYN DEV TYPE CHAR(3) REST OF CDYNWK 1ST WORD CITTRA FIXED(31) ALTERNATE ITTR PTR CDYNNUMB FIXED(31) NUMB OF DYN ALLOC DEVICES CITTR FIXED(31) ITTR PTR TO 1ST INP STRING CITTR FIXED(31) HI CONTROL FIELD FOR IMS CODIMSHI FIXED(16) HI CONTROL FIELD FOR IMS CODIMSHI FIXED(16) HI CONTROL FIELD LENGTH CODIMSHI FIXED(16) NUMBER OF E-PHASE BUFFERS CDEBUG CHAR(1) DEBUG SWITCH BYTE CEBUFFRS FIXED(16) NUMBER OF E-PHASE BUFFERS CDEBUG CHAR(1) NO = UNEXPECTED SYSIN EOF CEXTRACT PTR(31) ADDR. OF EXTRACT ROUTINE CEXPAND FIXED(31) INT LRECL EXPAND POS OR NEG CEMERGE FIXED(31) MERGE ORDER - ELIMINATION		2		FIYFD(31)	HIVIN INDEX COUNTED
CDEXPBUF FIXED(31) INDEX ENTRIES PER BUFFER CDEXPTRK FIXED(31) INDEX ENTRIES PER TRACK CDYNAMIC PTR(31) ADDR OF UNUSED DYNAMIC AREA CDYNWK CHAR(8) DYN ALLOC DEVICE TYPE CDYNWK1 PTR(8) 1ST BYTE OF DYN DEV TYPE CHAR(3) REST OF CDYNWK 1ST WORD CITTRA FIXED(31) ALTERNATE ITTR PTR CDYNNUMB FIXED(31) NUMB OF DYN ALLOC DEVICES CITTR FIXED(31) ITTR PTR TO 1ST INP STRING CITTRA FIXED(31) HI CONTROL FIELD FOR IMS CODIMSHI FIXED(16) HI CONTROL FIELD FOR IMS CODYNLEN PTR(8) DYNALLOC FIELD LENGTH CEBUFFRS FIXED(16) NUMBER OF E-PHASE BUFFERS CDEBUG CHAR(1) DEBUG SWITCH BYTE CENDMODE CHAR(1) NO = UNEXPECTED SYSIN EOF CEXTRACT PTR(31) ADDR. OF EXTRACT ROUTINE CEXPAND FIXED(31) INT LRECL EXPAND POS OR NEG CEMERGE FIXED(31) MERGE ORDER - ELIMINATION		2 .		PTR(31)	
CDYNWK CHAR(8) DYN ALLOC DEVICE TYPE  CDYNWK1 PTR(8) 1ST BYTE OF DYN DEV TYPE  CHAR(3) REST OF CDYNWK 1ST WORD  CITTRA FIXED(31) ALTERNATE ITTR PTR  CDYNNUMB FIXED(31) NUMB OF DYN ALLOC DEVICES  COTTR FIXED(31) ITTR PTR TO 1ST INP STRING  CDIMSHI FIXED(16) HI CONTROL FIELD FOR IMS  CDIMSHI FIXED(16) HI CONTROL FIELD LENGTH  COMPANIEN PTR(8) DYNALLOC FIELD LENGTH  CEBUFFRS FIXED(16) NUMBER OF E-PHASE BUFFERS  CDEBUG CHAR(1) DEBUG SWITCH BYTE  CENDMODE CHAR(1) NO = UNEXPECTED SYSIN EOF  CEXTRACT PTR(31) ADDR. OF EXTRACT ROUTINE  CEXPAND FIXED(31) INT LRECL EXPAND POS OR NEG  CEMERGE FIXED(31) MERGE ORDER - ELIMINATION		2	CDEXPBUF	FIXED(31)	INDEX ENTRIES PER BUFFER
CDYNWK CHAR(8) DYN ALLOC DEVICE TYPE  CDYNWK1 PTR(8) 1ST BYTE OF DYN DEV TYPE  CHAR(3) REST OF CDYNWK 1ST WORD  CITTRA FIXED(31) ALTERNATE ITTR PTR  CDYNNUMB FIXED(31) NUMB OF DYN ALLOC DEVICES  CITTR FIXED(31) ITTR PTR TO 1ST INP STRING  CDIMSHI FIXED(16) HI CONTROL FIELD FOR IMS  CDYNLEN PTR(8) DYNALLOC FIELD LENGTH  CEBUFFRS FIXED(16) NUMBER OF E-PHASE BUFFERS  CDEBUG CHAR(1) DEBUG SWITCH BYTE  CENDMODE CHAR(1) NO = UNEXPECTED SYSIN EOF  CEXTRACT PTR(31) ADDR. OF EXTRACT ROUTINE  CEXPAND FIXED(31) INT LRECL EXPAND POS OR NEG  CEMERGE FIXED(31) MERGE ORDER - ELIMINATION		2		FIXED(31)	INDEX ENTRIES PER TRACK
CDYNWK CHAR(8) DYN ALLOC DEVICE TYPE  CDYNWK1 PTR(8) 1ST BYTE OF DYN DEV TYPE  CHAR(3) REST OF CDYNWK 1ST WORD  CITTRA FIXED(31) ALTERNATE ITTR PTR  CDYNNUMB FIXED(31) NUMB OF DYN ALLOC DEVICES  CITTR FIXED(31) ITTR PTR TO 1ST INP STRING  CDIMSHI FIXED(16) HI CONTROL FIELD FOR IMS  CDYNLEN PTR(8) DYNALLOC FIELD LENGTH  CEBUFFRS FIXED(16) NUMBER OF E-PHASE BUFFERS  CDEBUG CHAR(1) DEBUG SWITCH BYTE  CENDMODE CHAR(1) NO = UNEXPECTED SYSIN EOF  CEXTRACT PTR(31) ADDR. OF EXTRACT ROUTINE  CEXPAND FIXED(31) INT LRECL EXPAND POS OR NEG  CEMERGE FIXED(31) MERGE ORDER - ELIMINATION	3	2			ADDR OF UNUSED DYNAMIC AREA
3 CDYNWK1 PTR(8) 1ST BYTE OF DYN DEV TYPE 3 * CHAR(3) REST OF CDYNWK 1ST WORD 3 CITTRA FIXED(31) ALTERNATE ITTR PTR 2 CDYNNUMB FIXED(31) NUMB OF DYN ALLOC DEVICES 3 CITTR FIXED(31) ITTR PTR TO 1ST INP STRING 2 CDIMSHI FIXED(16) HI CONTROL FIELD FOR IMS 2 CDYNLEN PTR(8) DYNALLOC FIELD LENGTH 2 CEBUFFRS FIXED(16) NUMBER OF E-PHASE BUFFERS 2 CDEBUG CHAR(1) DEBUG SWITCH BYTE 2 CENDMODE CHAR(1) NO = UNEXPECTED SYSIN EOF 2 CEXTRACT PTR(31) ADDR. OF EXTRACT ROUTINE 2 CEXPAND FIXED(31) INT LRECL EXPAND POS OR NEG 2 CEMERGE FIXED(31) MERGE ORDER - ELIMINATION	C	2	CDYNWK		DYN ALLOC DEVICE TYPE
3 * CHAR(3) REST OF CDYNWK 1ST WORD 3 CITTRA FIXED(31) ALTERNATE ITTR PTR 2 CDYNNUMB FIXED(31) NUMB OF DYN ALLOC DEVICES 3 CITTR FIXED(31) ITTR PTR TO 1ST INP STRING 2 CDIMSHI FIXED(16) HI CONTROL FIELD FOR IMS 2 CDYNLEN PTR(8) DYNALLOC FIELD LENGTH 2 CEBUFFRS FIXED(16) NUMBER OF E-PHASE BUFFERS 2 CDEBUG CHAR(1) DEBUG SWITCH BYTE 2 CENDMODE CHAR(1) NO = UNEXPECTED SYSIN EOF 2 CEXTRACT PTR(31) ADDR. OF EXTRACT ROUTINE 2 CEXPAND FIXED(31) INT LRECL EXPAND POS OR NEG 2 CEMERGE FIXED(31) MERGE ORDER - ELIMINATION	C	3			1ST BYTE OF DYN DEV TYPE
CDYNNUMB FIXED(31) NUMB OF DYN ALLOC DEVICES CITTR FIXED(31) ITTR PTR TO 1ST INP STRING CDIMSHI FIXED(16) HI CONTROL FIELD FOR IMS CDYNLEN PTR(8) DYNALLOC FIELD LENGTH CEBUFFRS FIXED(16) NUMBER OF E-PHASE BUFFERS CDEBUG CHAR(1) DEBUG SWITCH BYTE CENDMODE CHAR(1) NO = UNEXPECTED SYSIN EOF CEXTRACT PTR(31) ADDR. OF EXTRACT ROUTINE CEXPAND FIXED(31) INT LRECL EXPAND POS OR NEG CEMERGE FIXED(31) MERGE ORDER - ELIMINATION	_				
CITTR FIXED(31) ITTR PTR TO 1ST INP STRING CDIMSHI FIXED(16) HI CONTROL FIELD FOR IMS CDYNLEN PTR(8) DYNALLOC FIELD LENGTH CEBUFFRS FIXED(16) NUMBER OF E-PHASE BUFFERS CDEBUG CHAR(1) DEBUG SWITCH BYTE CENDMODE CHAR(1) NO = UNEXPECTED SYSIN EOF CEXTRACT PTR(31) ADDR. OF EXTRACT ROUTINE CEXPAND FIXED(31) INT LRECL EXPAND POS OR NEG CEMERGE FIXED(31) MERGE ORDER - ELIMINATION	0				
CDIMSHI FIXED(16) HI CONTROL FIELD FOR IMS CDYNLEN PTR(8) DYNALLOC FIELD LENGTH CEBUFFRS FIXED(16) NUMBER OF E-PHASE BUFFERS CDEBUG CHAR(1) DEBUG SWITCH BYTE CENDMODE CHAR(1) NO = UNEXPECTED SYSIN EOF CEXTRACT PTR(31) ADDR. OF EXTRACT ROUTINE CEXPAND FIXED(31) INT LRECL EXPAND POS OR NEG CEMERGE FIXED(31) MERGE ORDER - ELIMINATION	4	2			
CDYNLEN PTR(8) DYNALLOC FIELD LENGTH CEBUFFRS FIXED(16) NUMBER OF E-PHASE BUFFERS CDEBUG CHAR(1) DEBUG SWITCH BYTE CENDMODE CHAR(1) NO = UNEXPECTED SYSIN EOF CEXTRACT PTR(31) ADDR. OF EXTRACT ROUTINE CEXPAND FIXED(31) INT LRECL EXPAND POS OR NEG CEMERGE FIXED(31) MERGE ORDER - ELIMINATION	4 8				
CEBUFFRS FIXED(16) NUMBER OF E-PHASE BUFFERS CDEBUG CHAR(1) DEBUG SWITCH BYTE CENDMODE CHAR(1) NO = UNEXPECTED SYSIN EOF CEXTRACT PTR(31) ADDR. OF EXTRACT ROUTINE CEXPAND FIXED(31) INT LRECL EXPAND POS OR NEG CEMERGE FIXED(31) MERGE ORDER - ELIMINATION	В	2			
2 CEXTRACT PTR(31) ADDR. OF EXTRACT ROUTINE 2 CEXPAND FIXED(31) INT LRECL EXPAND POS OR NEG 2 CEMERGE FIXED(31) MERGE ORDER - ELIMINATION	C	2			
2 CEXTRACT PTR(31) ADDR. OF EXTRACT ROUTINE 2 CEXPAND FIXED(31) INT LRECL EXPAND POS OR NEG 2 CEMERGE FIXED(31) MERGE ORDER - ELIMINATION	Ĕ	- 2			
2 CEXTRACT PTR(31) ADDR. OF EXTRACT ROUTINE 2 CEXPAND FIXED(31) INT LRECL EXPAND POS OR NEG 2 CEMERGE FIXED(31) MERGE ORDER - ELIMINATION	F	2			
2 CEMERGE FIXED(31) MERGE ORDER - ELIMINATION	0	2			ADDR. OF EXTRACT ROUTINE
2 CEMERGE FIXED(31) MERGE ORDER - ELIMINATION	24	2			INT LRECL EXPAND POS OR NEG
2 CEXITPH1 FIXED(31) LNG OF PH1 RTNS		2		FIXED(31)	MERGE ORDER - ELIMINATION
		2	CEXITPH1	FIXED(31)	LNG OF PH1 RTNS

DISP LE	VEI LAR		 ATTP	COMMENT
230 2 234 2 238 2 238 2 23C 3 244 2 248 2 248 2 248 2 248 2 248 2 250 3	CEX CEX CE1 CEX CE1 4 CE1 4 CE1 4 CE1 4 CE3 5 CE3 6 CE1	ITPH2 ITPH3 8EOD IT11 1NAME 1LNG 1CALL IT15 5NAME 5ADDR 2MO 2MO 2CHOCH 5CALL	FIXED(31) FIXED(31) PTR(31) BDY(WORD) CHAR(8) FIXED(31) PTR(31) BDY(WORD) CHAR(8) PTR(31) FIXED(31) FIXED(31) FIXED(31) FIXED(31) PTR(24) FIXED(31)	COMMENT
254 2 3 3 2 2 5 6 C 2 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C 2 7 4 6 C	CE1 CE1 CEX	6NAME 6LNG 6CALL ITIT 7NAME 7LNG 7CALL ITI8 8NAME 8NAME 8LNG 8CALL	CHAR(8) FIXED(31) PTR(31) BDY(WORD) CHAR(8) FIXED(31) PTR(31) BDY(WORD) CHAR(8) FIXED(31) PTR(31)	NAME E16 LENGTH CALL ADDR E17 EXIT NAME E17 LENGTH CALL ADDRESS E18 EXIT NAME E18 LENGTH CALL ADDRESS
278 27C 280 284 284 28C 290 290 298 298 298 29C 2	CE1 CE1 CE2 CE3 CE1 CE1 CE3 CE6 CE6	8EXL 8PSW IT19 9NAME 9CALL IT61 INAME 1LNG 1CALL	PTR(31) PTR(31) BDY(WORD) CHAR(8) FIXED(31) PTR(31) BDY(WORD) CHAR(8) FIXED(31) PTR(31)	ADDR TO USER XLIST(5) RTN ADDR TO VSAM EXIT LIST ADDR TO VSAM PASSW. LIST E19 EXIT NAME E19 LENGTH CALL ADDRESS E61 EXIT NAME E61 LENGTH CALL ADDR E21 EXIT
29C 2A4 2A48 2A80 2B0 2B4 2B4 2BC 2BC	CE2 CE2 CEX	INAME ILNG ICALL IT27 ITAME ITAN ITAN ITAN ITAN ITAN ITAN INAME ILNG ILNG ICALL ICAL	CHAR(8) FIXED(31) PTR(31) CHAR(8) FIXED(31) PTR(31) BDY(WORD) CHAR(8) FIXED(31) PTR(31) PTR(31)	NAME E21 LENGTH CALL ADDR E27 EXIT NAME E27 LENGTH CALL ADDR E31 EXIT NAME E31 LENGTH CALL ADDR
2C8 3 2C8 2CC 2 2CC 3 2D4 3	CE3	5NAME 5ADDR 5LNG 5CALL IT37 7NAME 7LNG 7CALL IT38 8NAME 8LNG 8CALL IT39 9NAME 9LNG	CHAR(8) PTR(31) FIXED(31) PTR(31) BDY(WORD) CHAR(8) FIXED(31) PTR(31) BDY(WORD) CHAR(8) FIXED(31) PTR(31) BDY(WORD) CHAR(8) FIXED(31)	E35 EXIT - NAME PASSED ADDRESS LENGTH CALL ADDRESS CE37 EXIT NAME E37 LENGTH CALL ADDRESS E38 EXIT NAME E38 LENGTH CALL ADDRESS E39 EXIT NAME E39 LENGTH CALL ADDRESS E39 EXIT NAME CALL ADDRESS E39 EXIT NAME CALL ADDRESS

Licensed Material--Property of IBM
Page of LY33-8042-6, As Updated 31 March 1981, By TNL LN20-9345

DISP LEVEL	LABEL	ATTR	COMMENT
2F0 3 2F4 3	CE39EXL CE39PSW CE39XLST CE61PARM X CE61PCF CE61PADD	PTR(31)	ADDR TO VSAM EXIT LIST ADDR TO VSAM PASSW. LIST ADDR TO USER XLIST(5) RTN
2F8 3	CE39XLST	PTR(31)	ADDR TO USER YLIST(5) RIN
2FC 2	CE61PARM		E61 PARM LIST
2FF 3	X 05/1805	CHAR(3)	ZEROS
3	CEGIPUP	FTXFD(31)	REC ADDRESS
3	CE61PLEN	FIXED(31)	CONTR FIELD LEN
308 2 30C 2	CFIELDS	FIXED(31)	NUMBER OF CONTROL FIELDS
א מתצ	CFILSIZE	PTR(24)	CONTR FIELD NUMB REC ADDRESS CONTR FIELD LEN NUMBER OF CONTROL FIELDS USER SPECIFIED FILE SIZE THREE BYTES ADDR. OF FND OF BUFFER
310 2	CFILIMIT CFREECTR	PTR(31)	ADDR. OF END OF BUFFER
314 2 318 2	CFREECTR CGMINDEX	FIXED(31)	ADDR. OF END OF BUFFER AVAILABLE RSA BIN COUNT INDEX IN GETMAIN TABLE
31C 2	CMANMSG	FIXED(31)	INDEX IN GETMAIN TABLE PTR TO MSGTABLE FOR RESTORE SORT GENERATED DEFAULTS: MAXIMUM ALLOWED
320 2	CMANMSG CGENDOPT COREPARM	BDY(WORD)	SORT GENERATED DEFAULTS:
320 3	COREPARM *	FIXED(31)	MAXIMUM ALLOWED
321 4	CORFPM24	PTR(24)	FOR SORT/MERGE
. 324 3	CMAXLIM CMINLIM	FIXED(31)	MAXIMUM ALLOWED SIZE OF MAIN STORAGE FOR SORT/MERGE MAX LIMIT FOR SIZE(MAX) MIN VIRTUAL STRG FOR PEER SORT GEN DEFAULTS CONT:
328 3 32C 2	CMINLIM	FIXED(31)	MIN VIRTUAL STRG FOR PEER
32C 2 32C 3	CGENDOP1 CRESERVE		
700	×	CHAR(1)	RESERVED FOR
32D 4 330 3	CRSRVD24 CMSGOPTN	PTR(24)	CALLING PROGRAM
330 4	CCONSMOD	CHAR(1)	Y = MESSAGES TO CONSOLE
331 4	CPRTRMOD	CHAR(1)	Y = MESSAGES TO PRINTER
332 4 333 4	CPRINMOD CCRITMOD	CHAR(1)	Y = PRINT ALL MESSAGES Y = PPINT CPITTCAL MGC
334 3	CEQUALS	CHAR(1)	Y = PRESERV ORDER OF EQUALS
335 3 336 3	CLISTMOD	CHAR(1)	Y = LIST CONTROL CARDS
334 3 335 3 336 3 337 3 338 3 340 3 341 3 342 3 343 3 344 3 344 3 345 3 346 3	CERRU016 CERRINV	CHAR(1)	RESERVED FOR CALLING PROGRAM MESSAGE OPTION Y = MESSAGES TO CONSOLE Y = MESSAGES TO PRINTER Y = PRINT ALL MESSAGES Y = PRINT CRITICAL MSG Y = PRESERV ORDER OF EQUALS Y = LIST CONTROL CARDS Y = CRITICAL ERROR ABEND Y = CRITICAL ERROR ABEND NAME FOR PRINT DD CARD VIO UNDER MVS R=EXCPVR V=EXCP X=MVT A=ALL M=MAN N=NONE Y=RELEASE WORKSPACE Y=ALTOM. SEC. ALLOC. Y=VERIFY OUTPUT RECS Y=BLOCKSET MAY BE USED C=TRANSLATE CH AND AQ A=TRANSLATE AQ ONLY SVC INSTRUCTION
338 3	CERRINV CSYSNAME CVIO	CHAR(8)	NAME FOR PRINT DD CARD
340 3 341 3	CVIO	CHAR(1)	VIO UNDER MVS
342 3	CEXMODE CRESDNT CRELSE	CHAR(1)	A=ALL M=MAN N=NONE
343 3	CRELSE	CHAR(1)	Y=RELEASE WORKSPACE
344 3 345 3	CSECALC	CHAR(I)	Y=AUTOM. SEC. ALLOC.
346 3	CVERIFY CBLKSET	CHAR(1)	Y=BLOCKSET MAY BE USED
347 3	CCHALT	CHAR(1)	C=TRANSLATE CH AND AQ
348 3	CSVC	PTR(16)	A=TRANSLATE AQ ONLY SVC INSTRUCTION
34A 3	CCHECK	PTR(16) CHAR(1)	Y=CHECK COUNTERS
34B 3			N=COUNTER CHECK SUPPRESS
340 3	CSMF	CHAR(1)	S=SHORT SMF RECORD F=FULL SMF RECORD
34C 3	CVBLKSET	CHAR(1)	N =BYPASS VLR-BLOCKSET
34D 2	CSTAE		Y =VLR-BLOCKSET MAY BE USED
34E 2	CVIOREC	CHAR(1) CHAR(1)	Y = STAE WANTED AT ABEND VIO OR TAPE REC
34D 2 34E 2 2 352 2	*	CHAR(2)	RESERVED
352 2 354 2	CIDXPLB CGETMTMP	FIXED(15) PTR(31)	BLOCKS/LOGIC INDEX ADDR. OF TEMP. GETM. AREAS
354 3	CGETTMP	PTR(31)	ALTERNATE NAME
358 2	CGETREC	PTR(31)	ADDR. OF INPUT GET ROUTINE
358 3 35C 2	CTIOTIN CGETPRM	PTR(31) PTR(31)	PTR TO INPUT DD NAME (DEF) PERM AREAS CHAIN
360 2	CHECKDCB	FIXED(31)	CHECKPOINT DCB POINTER
364 2 368 2	CHKTTSTR	PTR(31)	CHKPT ITT SAVE AREA
36C 2	CHBUF1 CHKBLK	FIXED(31) FIXED(31)	INDEX HDR BFR ADDR TOTAL BLKS EST FOR PRIM EXT
370 2	CINSIZE	FIXED(31)	SORTIN LOGICAL RCD LENGTH
360 2 364 2 368 2 360 2 370 2 374 2 378 2	CIBUFFER CIBSPACE	PTR(31) FIXED(31)	ADDR. OF INPUT INDEX BUFFER INPUT BFR SPACE
37C 2	CINDEX	PTR(31)	ADDR. OF INPUT INDEX ENTRY
		• • •	and the second s

DISP	LEVEL	LABEL	ATTR	COMMENT
0488CC048CC048CC048C048C044688 C448C00048CE02448CCF4A 3383833333333333333333333333333333333	222 3 3 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3	CINDEXSIZ CINTERED CONTRIBLE CINTERED CONTRIBLE CONTRIBL	PTR(31) FIXED(31)	ADDR. OF END OF INDEX BUF INDEX ENTRY SIZE HEAD OF FREE TRACK HOLDERS SAVE INPUT PTR (ICELIM) HEAD OF AVAIL. TRACK QUEUE BLKSIZE OF 1-ST SORTIN FILE INDEX BLOCK SIZE ADDR TO INPUT REC SIZE OF CONTROL WORD TABLE FOR VARIABLE GETMAIN GETMAINED CHAIN SIZE OF RETURNED STORAGE BASE-BIN SIZE APTM411M LAST ENTRY TO REAL TIME RIN TIME ACCUMULATORS C-PHASE ELAPSED TIME C-PHASE CPU TIME P-PHASE CPU TIME R-PHASE CPU TIME R-PHASE CPU TIME R-PHASE CPU TIME R-PHASE CPU TIME SPECIAL CLOCK 0 SPECIAL CLOCK 1 SPECIAL CLOCK 1 SPECIAL CLOCK 5 NO. OF CLC-INSTR. TREEGEN INPUT BLOCKSIZE INPUT RECORD LENGTH LENGTH LAST CLC. TREEGEN INPUT RECORD LENGTH LENGTH LAST CLC. TREEGEN INPUT RECORD FORMAT NOT USED AFTER PH 0 CTL WORD MVC OR BAL TO RTN ADDR. OF TRMERGE ROUTINE TREEGEN WORK AREA DEBUG MESSAGE TYPE INDEX TRKS NOT TO RELEASE PARMS FOR VARIABLE GETMAIN MINIMUM SIZE WANTED SIZE OF WORKING STORAGE MINIMAL RECORD SIZE HALF-WORD MODAL RECORD MODAL RECORD MODAL RECORD MODAL RECORD MODAL RECORD
412 414 414 418 410 410 417 424	3	CMODULE CLINK CMODNAME CMOVERB(3)	CHAR(8) FIXED(31) CHAR(3) PTR(16) PTR(16) PTR(16) PTR(16) PTR(16)	MOD TO BE CALLED ZERO IF NO MORE MOD TO CALL SIGNIFICANT PART OF MODNAME RCD MVC RSA TO BUFFER

DISP LEVEL LABEL ATTR	COMMENT
46C 2 COUTLIM PTR(31) 470 2 COUNTERS 470 3 COUNTERO FIXED(31) 474 3 COUNTER1 FIXED(31) 478 3 COUNTER2 FIXED(31) 470 3 COUNTER3 FIXED(31) 480 3 COUNTER4 FIXED(31) 484 3 COUNTER5 FIXED(31) 488 3 COUNTER6 FIXED(31)	ADDR. OF END OF WORK BUF EVENT COUNTERS ENTRIES TO INOUTRIN ISSUED EXCPS RECORDS IN RECORDS OUT RECORDS OUT RECORDS DELETED SPECIAL COUNT 0 SPECIAL COUNT 1

DISP	LEVEL	LABEL	ATTR	COMMENT
4BC	2	CPRTDECB(5)	PTR(31)	PRINTER OUTPUT DECB END PTR TO UNPROT FIXD AREA ADDR. OF FIXED AREAS TABLE MIN. NO STRINGS ON SORTWK PAR INDEX BIN COMP PAR INDEX BIN MOVE RESERVED ********* (WORD 3) PRINT BUFFER VLR AND VLB INFORMATION (WORD 3) PRINTER OUTPUT PRINT LINE MSG VARIABLE INFO LIST
4 D O	2 2 2 2 2 2 2 2 2 3 3	CPAGELIM	PTR(31)	END PTR TO UNPROT FIXD AREA
4D4	2	CPAGELOC	PTR(31)	ADDR. OF FIXED AREAS TABLE
4D8	2	CPEERAGE	FIXED(31)	MIN. NO STRINGS ON SORTWK
4DC 4E2	2	CPARCLC(3)	PIR(16)	PAR INDEX BIN COMP
462	5	CPARMVC(3)	CHAR(2)	BECEBALD AAAAAAAAAA
4EA	2	CPRT	CHAR(129)	(WORD 3) PRINT BUFFER
4EA	_3	CPRTHEAD(4)	FIXED(15)	VLR AND VLB INFORMATION
4F2	3	CPRTLINE	CHAR(121)	(WORD 3) PRINTER OUTPUT
,	4	*	FIXED(15)	PRINT LINE
4F4	4	CMSGLIST(29)	CHAR(3)	REGION IS CORE LIMIT MAXIMUM G-VALUE - CREATION MAXIMUM G-VALUE - PARTTION MAXIMUM G-VALUE - REDUCTION MAXIMUM G-VALUE - REDUCTION MAXIMUM G-VALUE-ELIMINATION PHASE WORK AREA EXIT RETURN BRANCHTAB REC LENI IN DEC REC LEN2 IN DEC REC LEN3 IN DEC ADDR OF OUTPUT INDEX BFR ICEDEC - ICEDED COMM PTR TO DEV WORK AREA E15 PARAMEYER LIST E35 PARAMEYER LIST E35 PARAMETER LIST ADDR. OF OUTPUT PUT ROUTINE PTR OUTPT DD NAME (DEF-DEG) CORE VALUE FOR MSG 39 (DEV) PTR TO END OF LAST BUFFDO INDEX POOL ADDR PROT AREA SIZE PERM ALLOC SIZE REBUILD SPANNED RECORDS CORE USED FOR MSG 92/93 SIZE OF TRANSPOSED RECORD ADDR. OF RECORD - FOR VSAM  VLR PAD/RESTORE AREA RCD COUNT FOR CURRENT TRACK ADDR TO RECORD KEY SPANNED REC PTR
56B	4 2 2 2 2 2 2 2 3 4 4 4 4 4	CPREGION	CHAR(3)	PEGION IS COPE LIMIT
56C	2	CPHASECG	FIXED(31)	MAXIMUM G-VALUE - CREATION
570	Ž	CPHASEPG	FIXED(31)	MAXIMUM G-VALUE - PARTTION
574	2	CPHASERG	FIXED(31)	MAXIMUM G-VALUE - REDUCTION
578	2	CPHASEEG	FIXED(31)	MAXIMUM G-VALUE-ELIMINATION
57C	2	CPHASEWK	BDY(WORD)	PHASE WORK AREA
57C 57C	3,	CEXTILAB	CHARLIDI	EXII KEIUKN BKANCHIAB
580	4	CDECL 2	FIXED(31)	REC LEN2 IN DEC
584	4	CDECL3	FIXED(31)	REC LENS IN DEC
	4	COBUFFER	PTR(31)	ADDR OF OUTPUT INDEX BFR
588	4 5 6	CDECDED	PTR(31)	ICEDEC - ICEDED COMM
588	<sub>7</sub> 6	CDEVWK	PIR(31)	PIR 10 DEV WORK AREA
58C 590	3 3	CEISPARM	CHARCAS	EID PAKAMETER LIDI
59C	<b>ງ</b> ັ	CPUTREC	PTR(31)	ADDR. OF OUTPUT PUT ROUTINE
59C	2 3 3	CTIOTOUT	PTR(31)	PTR OUTPT DD NAME (DEF-DEG)
5 A O	3	CORMSG39	PTR(31)	CORE VALUE FOR MSG 39 (DEV)
5 A O	2 2 2 2 3	CPAGERSA	PTR(31)	PTR TO END OF LAST BUFFDO
5A4	2	CPBUF1	FIXED(31)	INDEX POOL ADDR
5A8 5AC	2	CPRUISIZ	LIXED(31)	PRUI AKEA 312E
5B0	2	CPERIITIN	PTR(31)	PERHITIN SPANNED PECOPOS
5B0	<b>-</b> 3	COREUSED	FIXED(31)	CORE USED FOR MSG 92/93
5B4	2 2 3	CRECSIZE	FIXED(31)	SIZE OF TRANSPOSED RECORD
5B8	2_	CRECADDR	PTR(31)	ADDR. OF RECORD - FOR VSAM
5B8	3	CSPNVSAM	PTR(31)	WIR DAR (DECTORE ADEA
5BC 5C0	2	CRECAREA	LIK(21)	VLK PAD/KESTUKE AKEA
5C4	2	CRECORD	FIXED(31)	ADDR TO RECORD KEY
5C8	2	CRECPTR	PTR(31)	SPANNED REC PTR
5CC	2	CRECPBUF	FIXED(31)	RECORDS/BUFFER
5D0	2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	CRECTBUF	FIXED(31)	ADDR TO RECORD KEY SPANNED REC PTR RECORDS/BUFFER RCD/BUF *TREE ENTRY SIZE(4) NUMBER OF R-PHASE BUFFERS RETURN CODE AREA (0 10) RECORD TYPE ON = FIXED LENGTH RECORDS
5D4 5D6	2	CRBUFFRS	FIXED(16)	NUMBER OF R-PHASE BUFFERS
5D7	2	CREICUDE	PTP(8)	RECORD TYPE
5D7	<b>-</b> 3	CFIXRFC	BIT(1)	ON = FIXED LENGTH RECORDS
5D7	3	CVARREC	BIT(1)	ON = VARIABLE LENGTH RECS
5D7	3	CSPANIN	BIT(1)	ON = VAR. SPANNED INPUT
5D8	2	CRESPMIM	PTR(31)	PERM RES CH OR TEMP RES
5DC 5E0	2	CRESTORE CRESTMP	PTR(31) PTR(31)	ADDR. OF RESTORE ROUTINE TEMP RESERVE CNT
5E4	2	CRLSE	CHAR(1)	YES FOR RELEASE OF SORTWK
5E5	2	CONSW3	PTR(8)	MISCELLANEOUS SWITCHES
5E5	3	CB37MSG	BIT(1)	B37 RECOVERY MAG FLAG aL5A
5E5	3 3	CWKVAMDS	BIT(1)	VIO WORK DATASET PRESENT
5E5 5E5	3 3	CSRTCHTL	BIT(1) BIT(1)	SORTCHTL STATEMENT PRESENT
5E5	3	CABRETRN CMERGEIN	BIT(1)	TO PREV SEC. ENTR OF ABENDR SORTINGS DD STMNT PRESENT
5E8		CRMERGE	FIXED(31)	MERGE ORDER - REDUCTION
5EC	2	CRPLIN	PTR(31)	ADDR OF VSAM INPUT RPL
5F0	2 2 2 2	CRPLOUT	PTR(31)	ADDR OF VSAM OUTPUT RPL
5F4	2	CRSAFREE	PTR(31)	ADDR. OF NEXT BIN IN RSA
5F4	3	CE35AREA	PTR(31)	

Licensed Material--Property of IBM
Page of LY33-8042-6, As Updated 31 March 1981, By TNL LN20-9345

DISP	LEVEL	LABEL	ATTR	COMMENT
E = 0		0040711	BWB4443	AVATU DAD DATUTE
5F8 5FC	2	CSYSIN	PTR(31)	SYSIN DCB POINTER
600	2	CSTSUUI	PIK(31)	STOUT DOB POINTER
604	2	COURTIN	LIVER(31)	SUBTUIT DOB FUTNIER
608	2	CSKIPREC	FIXED(31)	NIMRED OF DECOPOS TO SKIP
609	_3	CSKTPR24	PTR(24)	THREE BYTES
60C	2	CSCANLOC	PTR(31)	ADDR. OF LAST SCAN LOCATION
60C	3	CMAXREC	FIXED(31)	MAXIMUM RECORD SIZE
610	2	CSAVITTR	PTR(.31)	PREVIOUS INDEX BLOCK ITTR
614	2	CSTRINGS	FIXED(31)	STRING COUNT
618	2	CSELECT	PTR(31)	ADDR. OF TSELECT ROUTINE
61C	2	CSORTWK	PTR(31)	PTR TO SORTWK DCB LIST
620 680	2	CSTSUUID	PIK(31)	SYSUUI - DCB
682	2	COUDDITE	LIYED(12)	SUB-DIN SIZE
684	5	CSUBFBAS	LIVER(12)	SECUNDARY VITUCATION SHITCH
688	2	CSTAFSAV	PTR(31)	STAF WORK AREA -
***	_3	*	PTR(8)	SAVE ADDRESS -
689	3	CSTAES24	PTR(24)	FOR NEXT LEVEL
68C	2	CSECTRK	FIXED(31)	SEC.ALLOC. TRK COUNTER
690	2	CSAVEPTR	FIXED(31)	SAVE AREA POINTER
694	2	CSYSRES	FIXED(31)	SYSTEM NEEDS
698	2	CTREESIZ	FIXED(31)	SIZE OF ALLOCATED TREE
69C	2	CIRELUC	P1K(31)	ADDR. OF START OF TREE
6A0	۷,	CDIADENI	LIK(21)	BIRCATER TOYONTA THE FUIKT
6 Å 4	2	CTRESHID	FIXED(31)	FDFF RYTES IN DSA
6A8	2	CTMPSIZ	FIXED(31)	TEMP ALLOC SIZE
6AC	2	CTREE2	PTR(31)	LOOK AHEAD TREE
6B0	2	CTESTKEY(3)	PTR(16)	CLC FOR TWO CONTROL WORDS
6B8	2	CURRDECB	PTR(31)	ADDR. OF CURRENT DECB
6BC	2	CURRDATE	FIXED(31)	CURRENT DATE - PACKED DEC.
6C0 6C4	2	CURREC	PTR(31)	ADDR TO BUFFERED REC IN RSA
608	2	CVERACC	LIXED(21)	ACCUMULATOR FOR VEKIFY
600	2	CVSAMEVE	LIK(OI)	STODYCE DESENED EUD NOVM
6 D O	2	CVMOVEL	FIXED(31)	VIR - NO OF RYIES TO MOVE
6D4	Ž	CVREMSW	BIT(8)	ICEVRE MISC SWITCH
6 D4	_3	CINCORE	BIT(1)	1=INCORE SORT
6 D 4	3	CVIM	BIT(1)	1=BYPASS VED CALL VIM
6 D4	3	CE15DELT	BIT(1)	1=E15 DELETED
6 D4	3	CRSAFULL	BIT(1)	1=RSA IS FULL
6 D4	<u> </u>	CFRSTBIN	BIT(1)	1=FIRST BIN ALLOC
6 D 5	ູ້	CHIKAIAB	BII(I)	T-LINE D-DICK
6 D 6	2	CMKKITE	ETYED(15)	CIDI ELEID BYCED WIN DECIEN
6 D8	ž	CWTO	BDY(WORD)	WTO - MF=1
	_3	×	FIXED(15)	TEXT LENGTH
	3	×	PTR(16)	MCC FLAGS
6 DC	3	CMESSAGE	CHAR(120)	SYSIN DCB POINTER SYSOUT DCB POINTER SORTIN DCB POINTER SORTIN DCB POINTER NUMBER OF RECORDS TO SKIP THREE BYTES ADDR. OF LAST SCAN LOCATION MAXIMUM RECORD SIZE PREVIOUS INDEX BLOCK ITTR STRING COUNT ADDR. OF TSELECT ROUTINE PTR TO SORTWK DCB LIST SYSOUT - DCB SUB-BIN SIZE SUB-BIN SIZE SUB-BIN SIZE SUB-BIN SECONDARY ALLOCATION SWITCH STAE WORK AREA - SAVE ADDRESS - FOR NEXT LEVEL SEC.ALLOC. TRK COUNTER SAVE AREA POINTER SYSTEM NEEDS SIZE OF ALLOCATED TREE ADDR. OF START OF TREE PTR/DISPL TO UNIT TAB ENTRY BLKS/TRK LOACALLY FREE BYTES IN RSA TEMP ALLOC SIZE LOOK AHEAD TREE CLC FOR TWO CONTROL WORDS ADDR. OF CURRENT DECB CURRENT DATE - PACKED DEC. ADDR TO BUFFERED REC IN RSA ACCUMULATOR FOR VERIFY ADDR OF MODULE ICEERR STORAGE RESEVED FOR VSAM VLR - NO OF BYTES TO MOVE ICEVRE MISC SWITCH 1=NCORE SORT 1=BYPASS VED CALL VIM 1=E15 DELETED 1=RSA IS FULL 1=FIRST BIN ALLOC 1=FIRST SUB-BIN T=TAPE D=DISK CTRL FIELD BASED MIN RECLEN WTO - MF-L TEXT LENGTH MCC FLAGS MESSAGE AREA MESSAGE IDENTIFICATION MESSAGE NUMBER MESSAGE TAG
/ 55	4	X	CHAR(3)	MESSAGE IDENTIFICATION
6 DF	4	CMSGNO	CHAR(3)	MESSAGE NUMBER
6 E 2	4 4	CMSGTAG *	CHAR(1) CHAR(1)	MESSAGE TAG MESSAGE BLANK
6 E 4	4	CJOBNAME	CHAR(1)	JOB NAME
6 EC	4	CDOT	CHAR(1)	DOT
6 ED	4	CSTEPNAM	CHAR(8)	STEP NAME
6F5	4	CMSGBLNK	CHAR(1)	MESSAGE BLANK
6F6	_ 4	CMSGINFO	CHAR(94)	MESSAGE TEXT
	3	×	PTR(16)	DESCRIPTOR CODES
750		X	PTR(16)	ROUTING CODES
758 750	2	CWORKQ CWBPOOL	PTR(31)	ADDRESS OF WORK AREA QUEUE
760	2 2 2	CWBPOOLL	FIXED(31) FIXED(15)	WB-POOL ADDRESS WRITEBACK POOL SIZE
762	2	CWBPOOLC	FIXED(15)	CURRENT POOL SIZE
. · - <del>-</del>	_			

Licensed Material--Property of IBM Page of LY33-8042-6, As Updated 31 March 1981, By TNL LN20-9345

-	DISP LEVE	L LABEL	ATTR	COMMENT
	DI- 77668800 2 2222222222222222222222222222222	CWBPOOLR CWMINREC CEXIT25 CE25NAME CE25LNG CE25LNG CE25CALL CE25CALL CE25CALL CSYSLIN CLINDENT1 CLODCNT3 CINDEX1 CINDEX1 CINDEX1 CINDEX1 CINDEX1 CODECAREB CTANDEX1 CEXITLOA CONNEGE COUNT CENUGADOR CONDEGE CSYSLMOD CZONEGE CONEGE CONEGE	FIXED(15) FIXED(15) BDY(WORD) CHAR(8) FIXED(31) PTR(31) PTR(31) PTR(31) CHAR(1) CHAR(1) CHAR(1) PTR(31) CHAR(1)	OFFSET TO RESERVE BLOCK MIN EXTRACT IN NON-PEER EXIT25 EXIT E25 NAME EXIT E25 LENGTH AND ITS CALL ADDRESS SORTMODS DCB POINTER SYSLIN DCB POINTER SYSLIN DCB POINTER FLAGS LINKEDITING REQUEST LOAD COUNT PHASE 1 LOAD COUNT PHASE 2 LOAD COUNT PHASE 2 LOAD COUNT PHASE 3 BUFFER POINTERS  USED IN ICELINK POINTER IN SCANTAB EXIT LOAD ROUTINE ADDRESS EXIT DELETE ROUTINE ADDR. PICK UP EQUALS FIELD OP CODE ICM FOR STOW MACRO RECS DELETED BY E25 TEMP SAVE FOR CLISTMOD TO REMEMBER THAT WE LINK EXCP CAN BE USED FOR SORTIN EXCP -//- SORTOUT ICEBUG AREA ADDRESS SYSLMOD DCB ADDRESS SYSLMOD DCB ADDRESS SYSLMOD DCB ADDRESS SYSLMOD DCB ADDRESS INTERM. STOR. F. ZD W. E61 ADDRESS OF SORTIN OPEN EXIT ADDRESS OF SORTOUT -//- CANT USE EXCP FOR SORTOUT DISP IS MOD FOR SORTOUT MULTIVOLUME DATA SET FLAG EXCP OPEN EXIT ADDRESS ADDRESS OF EXCP DATA CODE MEMBER NAME FOR SORTIN SAVE AREA IN ALL PHASES CONCATENATED SORTIN FLAG PERFORM OPEN IN ICEOBS COUNT OF ACTION MESSAGES SMF WORK AREA PTR SMF STATISTICS RTN PTR
	804 2 808 2 80C 2 810 2 828 2	CFILBYTE CTOTTIME CAVGRCDL CTREEBAS CPATCH CTRACE	FIXED(31) FIXED(31) FIXED(31) FIXED(31) FIXED(31) CHAR(128)	OR EXTRACT RTN PTR FILE SIZE ACCUM (BYTES) CPU TIME AREA FOR SMF AVGERAGE RECORD LENGTH TREE BASECODE ADDRESS RESERVED TRACE TABLE

Licensed Material--Property of IBM
Page of LY33-8042-6, As Updated 31 March 1981, By TNL LN20-9345
This page intentionally left blank.

# Control Phase Information Area (CPI), Conventional Techniques

The program's Control Phase Information Area is set up by ICERCO and ICERCM. It is used for the communication of parameters that are needed by the general assignment routines. It is expanded into the PPI (Phase-to-Phase Information Area) by ICERC1 in the Optimization Phase.

	ACEMENT		FIELD	DESCRIPTION
DEC	(HEX)	SIZE	NAME	MEANING/USE
0	(0)	72	ICERC5	Register area
72	(48)	72		Control Field Information
72	(48)	2	CPINUMCF	No of control fields
74	(4A)	72	CPIPCFxx	12 entries in the form:
	• •	3		- Position
		2		- Length
		1		- Format + sequence
146	(92)	8	CPISW1	Switch of 64 Bits
				(See PPISW1 for details)
154	(9A)	4	CPILAB09	Output buffers:
			(Byte 1)	No of ph2 output buffers
			(Byte 2)	No of ph3 output buffers
			(Bytes 3-4)	Size of ph3 output buffers
158	(9E)	2	CPINWKU	No of work units
160	(A0)	2 2	CPIP1ASZ	Ph1 assignment size
162	(A2)	2	CPIIPBLK	Input blocking
164	(A4)	2	CPIFFF	Displacement of F field
466	<i>(</i> 2, <i>(</i> ),	•	08788W88	(used by extract routine)
166	(A6)	2	CPIPBUFF	Displacement of packing buffer
168	(A8)	2	CPIP2ASZ	Size of ph2 assignment modules
170 172	(AA)	2	CPIP3ASZ	Size of ph3 assignment modules
174	(AC) (AE)	2 2 2 2 2 2 2 2 2 2 2 2 3 3 3	CPIOPBLK CPIBUF23	Output blocking
176	(BO)	2	CPIBOF 25 CPISRTBL	No of buffers (ph2,3) B (blocking used on work areas)
178	(B2)	2	CPISKIBL CPIRCDL1	1, from RECORD stmt (LENGTH)
180	(B2) (B4)	2	CPIRCDL1	1 <sub>2</sub> from RECORD stat (LENGTH)
182	(B6)	2	CPIRCDL3	1 <sub>3</sub> from RECORD statt (LENGTH)
184	(B8)	2	CPIRCDL4	14 from RECORD stmt (LENGTH)
186	(BA)	2	CPIRCDL5	1, from RECORD stmt (LENGTH)
188	(BC)	2	CPILAB03	Size of ph1 input buffers
190	(BE)	2	CPILAB07	Output buffer information
192	(CO)	2	CPIPGCSZ	Size of area used by extract rtn
194	(C2)	2	CPIBINSZ	Bin size
196	(C4)	3	CPISKPRD	Count of skipped records
199	(C7)	3	CPIATP1E	Addr of invoking pgm's ph1 exit
202	(CA)	3	CPIMODEX	Exits Activated
				(see first 3 bytes of PPIMODEX
				for details)
205	(CD)	3	CPIATP3E	Addr of invoking pgm's ph3 exit
208	(D0)	3	CPITAVLC	Size of available core, ph1
211	(D3)	3	CPIFILSZ	Value of FILSZ/SIZE parameter
214	(D6)	3	CPISRTG	G (Capacity of RSA, in records)
217	(D9)	3 3	CPISIRG	Size of interblock gap (tape wk)
220	(DC)	<b>3</b>	CPIP1GC	Size of ph1 work area
220 223	(DC)	3 3 3	CPIP1RSZ CPIP2GC	Ph1 running modules size Size of ph2 work area
223	(DF) (DF)	3	CPIP2GC CPIP2RSZ	Ph2 running modules size
226	(E2)	3	CPIP2RSZ CPIP3GC	Size of ph3 work area
226	(E2)	3	CPIP3GC CPIP3RSZ	Ph3 running modules size
229	(E5)	1	CPIBUF1	No of ph1 buffers
230	(E6)	1	CPIMRGMX	Maximum merge order
	<b></b> ,	•		

DISPLACEMENT			FIELD	DESCRIPTION
DEC	(HEX)	SIZE	NAME	MEANING/USE
231	(E7)	1	CPIMRGAL	Alternate merge order
232	(E8)	3	CPILINK	Link Edit Info for exit rtns
	-	XXXX XXXX	(Byte 1)	Exits in order as for PPIMODEX:
		XXXX XXXX		1=Routine was link edited,
		x	(Byte 3)	0=Not edited (via the program)
		.1		E11 routine edited separately
		1		E21 routine edited separately
		1		E31 routine edited separately
		1		Linkage editing was carried out
		1		VSAM output not reusable
		1.		Linkage edit error
		1		VSAM release 1 in system
235	(EB)	3	CPIADDCF	When more than 12 ctl fields:
				addr of additional information
238	(EE)	8	CPISORCE	DDname of user mod library
246	(F6)	4	CPIDDSRT	4 chars to replace SORT in DDname
				(fr param list when pgm invoked)
250	(FA)	2	CPIDCBIN	Size of SORTIN DCB
252	(FC)	2	CPIDCBOU	Size of SORTOUT DCB
254	(FE)	4	CPIXCAP	OSCL:RMAX (max no of bytes/reel)
				Otherwise: Capacity of interm.
				storage (in records)
258	(102)	16	CPIDEV	Device inf: words 2-5 fr DEVTYPE
		_		macro (used by ICERCI)
274	(112)	3	CPIAUENT	Addr of user exit name table
277	(115)	1	CPISUENT	Size of user exit name table
278 206	(116)	8	CPIMSGDD	Message DDname if not SYSOUT
286	(11E)	1	CPIRECFM	User record format for ph3
287	(11F)	8	CPIJOBNM	Johname
295 203	(127)	8 1	CPISTPNM	Stepname
303	(12F)	_	CPISW2 CPIFSEON	Switch of 8 Bits
		1	CPIPSEON	FILSZ given with exact value Exec sort with ABEND option
		.1		AQ control fields present
		1		Preserve order of equals
				VSAM input
		1		KSDS VSAM output
		1		ESDS VSAM output
		1		SIZE (MAX) used
306	(130)	2	CPIMXCOL	Rightmost col included in any
J J J	(130)	L	CETHINCOL	control field
308	(132)	1	CPIDEVCD	Device code for unit with RPS
309	(133)	256	CPIAOTAB	Translate table for use with
	(,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	277	CETUATUD	AAGUUAGIG CODAG LUA UOG WALII

# Phase-to-Phase Information Area (PPI), Conventional Techniques

The Phase-to-Phase information area (PPI) is a communication area for all modules of the conventional sort/merge program. It is created during Phase 0 and resides in main storage throughout execution. It is not executable, and has the CSECT name ICERCA.

The program uses register 13 as the base register for the PPI. Reference to any field within the area can be made by adding the appropriate displacement to the contents of register 13. The displacement for each field within the PPI appears in decimal and hexadecimal form in the description which follows Figure 15.

# Abbreviations

Mult means a multiple compare routine is used to compare records;

No mult means it is not;

Extract means an extract routine is used to compare records.

means records are fixed length; Var means records are variable length.

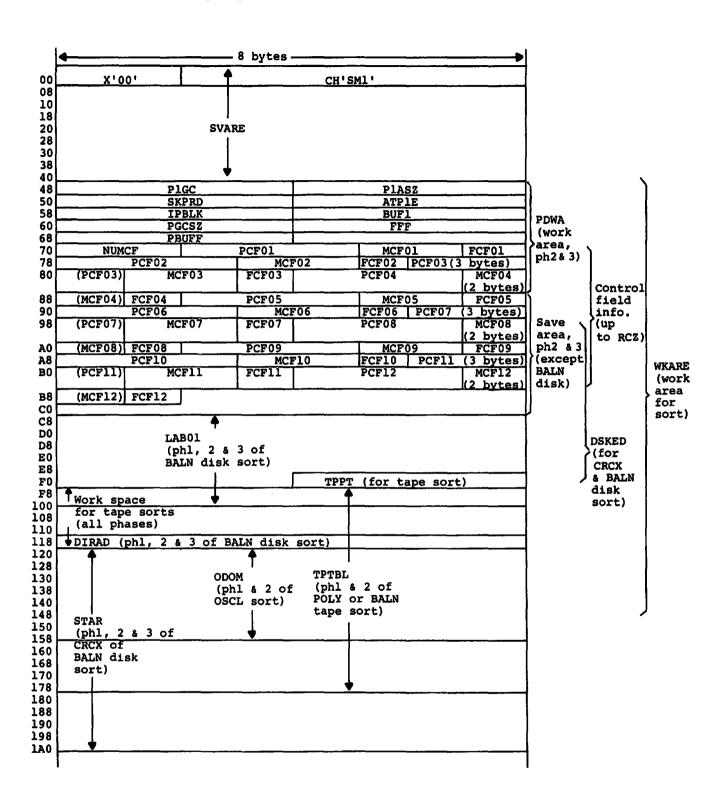


Figure 15. (1 of 3) Map of PPI

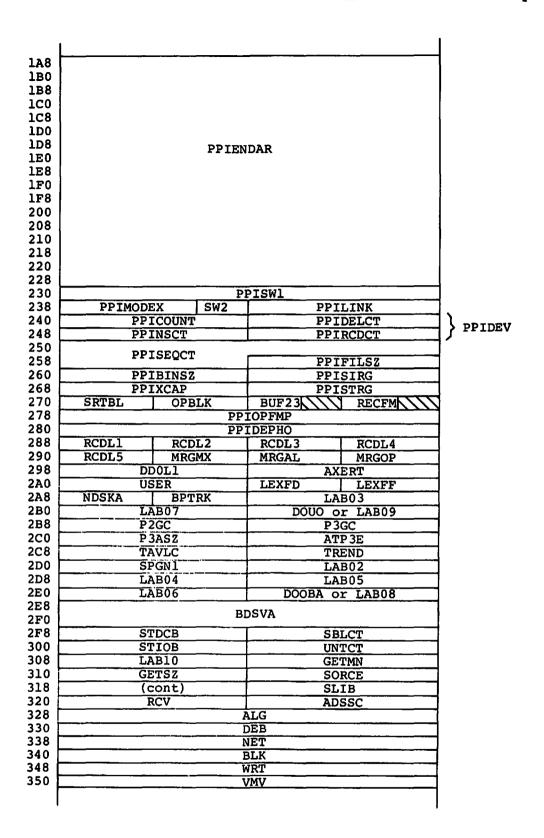


Figure 15. (2 of 3) Map of PPI

358			R			
360			DE			
368			NE			
370			BL.			
378			N,			
380			CO			
388 390				OF		
390 398				MA		
390 3A0				r RMB MA		
3AU				r AMB	_	
3B0			OP:			
3B8				11		
3C0	X31 or X21					
3C8		X15 or E32				
3D0		X35 or X25				
3D8	<del></del>	X17				
3E0		X37 or X27				
3E8				18		
3F0		X38 or X28				
3F8		X19				
400				r X29		
408				61		
410			X	16		
418		ADDCF			DDS	SRT
420		mm	IIII		CHI	KAD
428	DCBIN	DCBC	Ū	TPCYL		DPTRK
430	CPTRK		ARGC		777	
438		NETAR			FT?	TAB
440		JOBNM				
448			ST	PNM		
450	MXCOL	DEVCD			OPI	
458		MOVEQ		RPLDP		VSMSL
460		VSMSG			VS	3I
						ĺ

Figure 15. (3 of 3) Map of FPI

DISPL	ACEMENT		FIELD	DESCRIPTION: CONTENTS,
DEC	(HEX)	SIZE	NAME	MEANING/USE
0	(0)	1		X'00'
1	(1)	3	PPISM	CH'SM1' (program identifier)
72	(48)	256	PPIWKARE	Program work area (sort)
72	(48)	64	PPI PDWA	program work area (merge)
72	(48)	4	PPI P 1GC	Size of Ph1 work area
76	(4C)	4	PPI P 1ASZ	Ph1 assignment size
80	(50)	4	PPISKPRD	Count of skipped records
84	(54)	4	PPIATP 1E	Address of invoking program's
				Ph1 exit
88	(58)	4	PPIIPBLK	Input blocking
92	(5C)	4	PPIBUF1	No of Ph1 buffers
96	(60)	4	PPIPGCSZ	Size of area used by extract
400		44	22222	routine
100	(64)	4	PPIFFF	Displacement of F field (used by extract routine)
400	46.01	4		Displacement of packing buffer
104	(68)	4	PPIPBUFF	Reserved
108	(6C)	4		Control Field Information
112	(70) (70)	72	PPINUMCF	No of control fields
112 114	(70) (72)	2 3	PPINOMER PPIPCF01	Control field 1: - Position
117	(72) (75)	2	PPIMCF01	- Length
119	(75) (77)	1	PPIFCF01	- Format + sequence
120	(77) (78)	3	PPIPCF02	Control field 2: - Position
123	(76) (7B)	3 2	PPIMCF02	- Length
125	(7D)	1	PPIFCF02	- Format + sequence
126	(7E)	ġ	PPIPCF03	Control field 3: - Position
129	(81)	3 2	PPIMCF03	- Length
131	(83)	1	PPIFCF03	- Format + sequence
132	(84)	3	PPIPCF04	Control field 4: - Position
135	(87)	2	PPIMCF04	- Length
136	(88)	64	<b>PPI PSVA</b>	Merge network prime area
137	(89)	1	PPIFCF04	- Format + sequence
138	(8A)	3	PPIPCF05	Control field 5: - Position
141	(8D)	3 2 1	PPIMCF05	- Length
143	(8F)		PPIFCF05	- Format + sequence
144	(90)	3	PPIPCF06	Control field 6: - Position
147	(93)	2	PPIMCF06	- Length
149	(95)	1	PPIFCF06	- Format + sequence
150	(96)	3	PPIPCF07	Control field 7: - Position
152	(98)	48	PPIDSKED	Used for up to 17 disk
				addresses, Ph1,2,3. Format:
				Assignment mod: ABCC
				(A=channel address
				B=unit address CC=no of tracks)
				Running mod: M BB CC HH R
453	(00)	2	DDTWCEA7	Control field 7: length
153 155	(99)	2 1	PPIMCF07 PPIFCF07	- Format + sequence
156	(9B) (9C)	3	PPIPCF08	Control field 8: - Position
159	(9F)	2	PPIMCF08	- Length
161	(A1)	1	PPIFCF08	- Format + sequence
162	(A2)	3	PPIPCF09	Control field 9: - Position
165	(A5)	3 2	PPIMCF09	- Length
167	(A7)	ĩ	PPIFCF09	- Format + sequence
168	(A8)	3	PPIPCF10	Control field 10: - Position
171	(AB)	3 2	PPIMCF10	- Length
173	(AD)	1	PPIFCF 10	- Format + sequence
174	(AE)	3 2	PPIPCF11	Control field 11: - Position
177	(B1)		PPIMCF11	- Length
179	(B3)	1	PPIFCF11	- Format + sequence

DISPLA	CEMENT		FIELD	DESCRIPTION: CONTENTS,
DEC	(HEX)	SIZE	NAME	MEANING/USE
		_		
180	(B4)	3	PPIPCF12	Control field 12: - Position
183	(B7)	2	PPIMCF12	- Length
185	(B9)	1	PPIFCF 12	- Format + sequence
186	(BA)	14		Reserved
200	(C8)	64	PPILAB01	General work area
				(used mainly by read/write)
244	(F4)	4	PPITPPT	Tape table pointer
248	(F8)	136	PPITPTBL	Tape Table
				Two-byte entry for each unit:
		1 0000	(Byte 1)	Input (0=output)
		.1 0000		Open routine should open unit
		1. 0000		OSCL: unit contains full reel
		1 0000		OSCL: descending sequence
				(set at EOF or RMAX)
		XXXX XXXX	(Byte 2)	DCB increment
280	(118)	8	PPIDIRAD	Disk directory address
288	(120)	64	PPIODOM	OSCL: odometer table
200	(120)	• • • • • • • • • • • • • • • • • • • •	1110001.	Disk starting addresses
288	(120)	136	PPISTAR	- one entry for each
200	(120)			of three to 17 extents
424	(1A8)	32	PPIENDAR	BALN (disk) & CRCX (Ph0,1,2):
	()		T T T TIME TIME	end addr reset from PPIDSKED
552	(228)	8		CRCX (Ph3): disk start addr
332	(220)	· ·		(for read priming)
560	(230)	8	PPISW1	Switch of 64 Bits
500	(230)	1	(Byte 1)	Fix
		.1	(D) CC ()	Var
		1		No mult.
		1		Mult.
		1		BALN (tape or disk)
		1		POLY
		1.		OSCL
		X		Reserved
		X	(Byte 2)	Reserved
		.1	(D) cc a)	Tape work areas
		1		BALN (disk)
		1		No data chaining
		1		Data chaining, input
		1		Data chaining, input  Data chaining, output
		1.		No exits to be used
				Exits to be used
		1	(Byte 3)	No records over 256 bytes
		.1	(D) CE 3)	Records over 256 bytes
		1		SKIPREC option used
		1		Ph1 in progress or complete
		1		Ph2 in progress or complete
		1		Ph3 (sort) in prog or complete
		1.		Ph3 (merge) in prog or compl
		1		Checkpoints to be taken
		1	(Byte 4)	Mult.
		.1	(D) ce -1	Extract
		1		SORTOUT in descending order
				(0=ascending)
		1		Order in Ph1 (or merge input)
				is descending (0=ascending)
				and and to appoint this

ISPLACEME EC (HE		FIELD NAME	DESCRIPTION: CONTENTS, MEANING/USE
	1		Descending order for merge pass (0=ascending)
	1		SM1 invoked (0=executed) FILSZ or SIZE given
	1		Merge-only: input present
			OSCL: QSAM has detected EOF
	1	(Byte 5)	'E'-type ctl fields present
	.1		SORT statement present MERGE statement present
	1		RECORD statement present
			(or, from ICERCI onwards, dis
	4		work devices with RPS)
	1		MODS statement present RECORD stmt absent or wrong
	1.		Buffers doubleword aligned
	1		Buffers fullword aligned
	1	(Byte 6)	Read error Write error
	1		BALN (tape): odd-numbered pas
			(0=even)
	x x		Channels available:
			00 - one multiplexor 01 - one multiplexor and/or
			one selector
			<pre>10 - one multiplexor and/or</pre>
	4		several selector
	1		Input unit used as work unit Tape switch or alternative
			channel available
	1		More than 1 channel available
	1	(Byte 7)	Deblock backward Read forward
	1		Close with rewind
	1		Block forward
	1		Read forward later
	x		Reserved CRCX
	1		DIAG option being used
	1	(Byte 8)	OSCL: user EOF
	.1		OSCL: RMAX reached
			OSCL: user insert in process Some or all tapes are 7-track
	1		CRCX: merge pass to follow
	1		Sort called by IMS
	1.		Up to ICERCN: max. no. of control fields present
			from ICERCN: BALN
	1		User rtn has given
68 (23)	D) 2	DDTMANDV	EROPT=skip/accept
68 (23)	B) 3 1	PPIMODEX	Exits Activated E11
	.1		E15 or E32
	1,		E16
	1		E17 E18
	1		E 10 E 2 1
	1.		E25
	4		<b>E27</b>

DISPL DEC	ACEMENT (HEX)	SIZE	FIELD NAME	DESCRIPTION: CONTENTS, MEANING/USE
		1	(Buto 2)	E28
		.1	(Byte 2)	E31
		1		E35
		1		E37
		1		E38
		1		E61
		1.		E 19
		1		E29
		1	(Byte 3)	E39
		.xxx x		Reserved
		1		RECFM=FB in SYSOUT DCB
		••••		Input is spanned records
571	/22D)	1	PPISW2	Output is spanned records Switch of 8 Bits
371	(23B)	1	PPISW2 PPIFSEON	FILSZ given with exact value
		.1	PPIP SEON	Exec sort with ABEND option
		1		AQ control fields present
		1		Preserve order of equals
		1		VSAM input
		1		KSDS VSAM output
		1.		ESDS VSAM output
				SIZE (MAX) used
572	(23C)	4	PPILINK	Link Edit Information
				for exit rtns (set by RCH)
		XXXX XXXX	(Byte 1)	Exits (same order as PPIMODEX):
		XXXX XXXX	(Byte 2)	1=Routine link edited,
		X	(Byte 3)	<pre>0=Not edited (via the program) E11 routine edited separately</pre>
		1		E21 routine edited separately
		1		E31 routine edited separately
		1		Linkage editing carried out
		1		VSAM output not reusable
		1.		Linkage edit error
		1		VSAM release 1 in system
		XXXX XXXX	(Byte 4)	Reserved
576	(240)	16	PPIDEV	Device inf: words 2-5 fr DEVTYP
576	(200)	44	PPTGGING	macro used by ICERCI
576	(240)	4	PPICOUNT	Count of records:
				Ph1 - deblocked from input Ph2 - written to work units
				Ph3 - placed on SORTOUT
580	(244)	4	PPIDELCT	Count of records deleted
584	(248)	4	PPIINSCT	Count of records inserted
588	(24C)	4	PPIRCDCT	Count of records entering ph.
592	(250)	12	PPISEQCT	BALN (disk): count of strings
				per work area (halfword/area)
				BALN (tape): count of strings-
			(Bytes 1-4)	- entering phase
604	125C	4	(Bytes 5-8)	- written out
608	(25C) (260)	4	PPIFILSZ PPIBINSZ	Value of FILSZ or SIZE param. Bin size
612	(264)	4	PPISIRG	Size of interblock gap (tapewk)
612	(264)	4	PPISRALG	Addr of sector conversion rtn
	,·,	•		(RPS)
616	(268)	4	PPIXCAP	OSCL: RMAX (max no. bytes/reel)
	• •			Otherwise: Cap. of interm.
				storage, in records
620	(26C)	4	PPISRTG	G (Cap. of RSA, in records)
624	(270)	2	PPISRTBL	B (blocking used on wk areas)
626	(272)	2	PPIOPBLK	Output blocking

DT000				DECENTANTON - COMPUSE
	LACEMENT		FIELD	DESCRIPTION: CONTENTS,
DEC	(HEX)	SIZE	NAME	MEANING/USE
628	(274)	2	PPIBUF23	No of buffers (Ph2,3)
630	(276)	2 1	PPIRECFM	User record format for Ph3
631	(277)	i	PPIRECPM	Reserved
632		8	DDTADWD	Output unit for Ph3
640	(278)	0	PPIOPFMP	Output unit address
648	(280)	0	PPIDEPHO	1 <sub>4</sub> from RECORD stmt (LENGTH)
650	(288)	2	PPIRCDL1	
	(28A)	2	PPIRCDL2	1 <sub>2</sub> from RECORD stmt (LENGTH)
652	(28C)	2	PPIRCDL3	1 <sub>3</sub> from RECORD stmt (LENGTH) 14 from RECORD stmt (LENGTH)
654	(28E)	2	PPIRCDL4	
	(290)	8 2 2 2 2 2 2	PPIRCDL5	15 from RECORD stmt (LENGTH)
658	(292)	2	PPIMRGMX	Maximum merge order
660	(294)	2	PPIMRGAL	BALN & POLY: alternate merge order
662	(296)	2	DDTMDCOD	Disk sort: optimum merge order
		4	PPIMRGOP	
664	(298)	4	PPIDDOL1	Ph2,3: length & offset of control field 1
660	(200)	41	0073 vmnm	
668	(29C)	4	PPIAXERT	Addr of mult. or extract rtn
672	(300)	4	PPIUSER	User communication area
676	(2A4)	2 2 2 2	PPILEXFD	Length of extracted fields
678	(2A5)	2	PPILEXFF	PPILEXFD rounded up to fullwd
680	(2A8)	2	PPINDSKA	No of disk work areas
682	(2AA)	2	PPIBPTRK	Blocks/track for disk wk areas
684	(2AC)	4	PPILAB03	Input buffer information:
			(Byte 1)	No of buffers
			(Bytes 2-4)	Size of buffers
688	(2B0)	4	PPILAB07	Output buffer information:
			(Byte 1)	No of buffers
			(Bytes 2-4)	Size of buffers
692	(2B4)	4	PPIDOUO	X'04'=suppress sequence check
				(passed by user from E35)
692	(2B4)	4	PPILAB09	Used to alloc. output buffers:
			(Byte 1)	No of Ph2 output buffers
			(Byte 2)	No of Ph3 output buffers
			(Bytes 3-4)	Size of Ph3 output buffers
696	(2B8)	4	PPIP2GC	Size of Ph2 work area
700	(2BC)	4	PPIP3GC	Size of Ph3 work area
704	(2C0)	4	PPIP3ASZ	Message option
708	(2C4)	4	PPIATP3E	Addr of invoking pgm's Ph3 rtn
712	(2C8)	4	PPITAVLC	Size of available core, Ph1
716	(2CC)	4	PPITREND	Ending address of tree
720	(2D0)	4	PPISPGN1	Address of next byte available
				in phase work area
724	(2D4)	4	PPILAB02	Address of input buffer table
	•			OSCL: address of Ph1 input
				buffer table
728	(2D8)	4	PPILAB04	Address of output buffer 1
732	(2DC)	4	PPILAB05	Address of output buffer 2
736	(2E0)	4	PPILAB06	Address of control buffer -
				Ph1: input buffer pool
				Merge-only Ph3: out.buff. pool
740	(2E4)	4	PPIDOOBA	Addr of record for Ph3
	- <b>*</b>			sequence check
740	(2E4)	4	PPILAB08	RSA Table information:
	• •		(Byte 1)	No of entries
			(Bytes 2-4)	Address of table
744	(2E8)	16	PPIBDSVA	Block/deblock area:
			(Byte 1)	No of work units
			(Bytes 2-4)	Addr of inp.buff. table, Ph2&3
			(Bytes 5-8)	Fix: addr of Ph1 move list
				Var: addr of next avail. bin
				<del></del>

DISPL DEC	ACEMENT (HEX)	SIZE	FIELD NAME	DESCRIPTION: CONTENTS, MEANING/USE
				*******************
			(Bytes 9-12)	Fix: addr of Ph2&3 move list
			muhaa 12 169	Var: no of available bins
760	/2mgs	4	(Bytes 13-16)	Var: entry to Ph1 move rtn Address of DCB table
764	(2F8) (2FC)	4	PPISTDCB PPISBLCT	Address of block count table
768	(300)	4	PPISTIOB	Address of IOB table
772	(304)	4	PPIUNTCT	OSCL: addr of unit count table
776	(308)	ů.	PPILAB10	OSCL, Ph2: addr of input buffer
, , ,	(000)	•		table
780	(30C)	4	PPIGETMN	Addr of GETMAIN table of addr.
784	(310)	4	PPIGETSZ	Addr of GETMAIN table of sizes
788	(314)	8	PPISORCE	Up to RCZ: DDname of user mod
				library
				RCZ: DCB addr of SYSLMOD user
				user lib (bytes 1-4)
796	(31C)	4	PPISLIB	DCB address of sort library
800	(320)	4	PPIRCV	Reserved
804	(324)	4	PPIADSSC	Return address to ICERCV
808	(328)	240	22717	Module Interface List
808 816	(328)	8 8	PPIALG	Ph 1, 2: Algorithm Ph 1, 3: Deblock
824	(330) (338)	0	PPIDEB PPINET	Ph1,3: Deblock Ph1,3: Network
832	(340)	8 8	PPIBLK	Ph1,3: Network Ph1,3: Block
840	(348)	8	PPIWRT	Ph1,2: Write
848	(350)	8 8	PPIVMV	Ph1: Move (var)
856	(358)	8	PPIRD	Ph2,3: Read
864	(360)	8	PPIDEB2	Ph2: Deblock (prime rtn)
872	(368)	8 8	PPINETM	Ph2: Network
880	(370)	8 8	PPIBLK2	Ph2: Block/deblock
888	(378)	8	PPIINT	OSCL: Initialize sort and tree
896	(380)	8	PPICONV	Convert hex to char for msgs
904	(388)	8	PPIEOF	Ph1,3 (merge-only): EODAD for
		•		QSAM
912	(390)	8	PPIRMA	Messages, Ph1 running modules
920	(398)	8	PPIRMB	Messages, Ph2 running modules Messages, Ph3 running modules
920 928	(398) (3A0)	8 8	PPIRMC PPIAMA	Messages, Ph1 assignmt modules
936	(3A8)	8	PPIAMB	Messages, Ph2 assignmt modules
936	(3A8)	8	PPIAMC	Messages, Ph3 assignmt modules
944	(3B0)	8	PPIOPEN	Ph1,2,3: Open list
952	(3B8)	8	PPIX 11	Exits for
960	(3C0)	8	PPIX21	functions
960	(3C0)	8	PPIX31	user initialization
968	(3C8)	8	PPIX151X32	Exits for
976	(3D0)	8	PPIX25	modification
976	(3D0)	8	PPIX35	logical record
984	(3D8)	8	PPIX 17	Exits for
992	(3E0)	8	PPIX 27	at end of phase
992	(3E0)	8 8	PPIX37	closing user data sets Exits
1000 1008	(3E8) (3F0)	8	PPIX18 PPIX28	errors
1008	(3F0)	8	PPIX 38	for read
1016	(3F8)	8	PPIX 19	Exits
1024	(400)	8	PPIX29	errors
1024	(400)	8	PPIX39	for write
1032	(408)	8	PPIX61	Exit to modify control fields
1040	(410)	8	PPIX 16	Exit when NMAX exceeded
1048	(418)	4	PPIADDCF	When more than 12 ctl fields:
				addr of additional information

DISPL	ACEMENT		FIELD	DESCRIPTION: CONTENTS,
DEC	(HEX)	SIZE	NAME	MEANING/USE
1052	(41C)	4	PPIDDSRT	4 chars to replace SORT in
	, ,			DDnames (from param list when
4056	44201	4		the program invoked) PPI version number
1056	(420)	1		
1057	(421)	3		Reserved
1060	(424)	4 2 2 2 2	PPICHKAD	Checkpoint module address
1064	(428)	2	PPIDCBIN	Size of SORTIN DCB
1066	(42A)	2	PPIDCBOU	Size of SORTOUT DCB
	(42C)	2	PPITPCYL	No of trks/cyl. (disk wk areas)
1070	(42E)	2	PPIDPTRK	No of directory blocks/track (disk work areas)
1072	(430)	2	PPICPTRK	Track capacity in bytes
1074	(432)	3	PPIAPGC	Addr of permanently gen. core
1077	(435)	2 3 3		Reserved
1080	(438)	4	PPINETAR	Addr of network table in phase
	( ,	•		work area
1084	(43C)	4	PPIFTTAB	BALN (disk): pointer to table
	( ,	•		of formatted tracks
1088	(440)	8	PPIJOBNM	Johname
1096	(448)	8	PPISTPNM	Stepname
1104	(450)	8 2	PPIMXCOL	Rightmost col included in any
	(450)	•	112.0.00	control field
1106	(452)	1	PPIDEVCD	Device code for unit with RPS
1107		i		Reserved
1108	(454)	ų.	PPIOPBAD	Addr of record in output buffer
	(454)	•	IIIOIDAD	for Ph2 sequence check
1112	(458)	4	PPIMOVEQ	Addr of 2nd move rtn for equals
	(45C)	2	PPIRPLDP	RPL displ. from ACB
	(45E)	2 2 4	PPIVSMSL	Length of generated msg area
1120	(460)	L L	PPIVSMSG	Addr of general VSAM msg area
1124	(464)	4	PPIVSI	Ph1 addr of VSAM read module
1124	(404)	4	<b>LLTADT</b>	Fire addit of vamiliand module

# Index to PPI

The following index covers PPI fields only. For any given subject, it shows the name of the PPI field referenced, plus the field's displacement from the beginning of the PPI area, in hexadecimal.

Since all fields begin with the letters 'PPI', these letters are omitted.

When the reference is to a field which occupies a whole number of bytes, the displacement is shown in whole bytes. When the reference is to one or more bits within the field, the reference is given in the form 'bytes.bits'. For example, the field PPISW1 is at displacement 560. A reference to 'SW1,561.0' therefore means the first bit (bit 0) of the second byte in the field.

Ascending	- extract techn used SW1,562.?
- order for SORTOUT SW1,563	3.2 (see also F field)
- order for merge pass SW1,50	
Assignment modules	- addr of 2nd move rtn
- size in ph1 P1ASZ,	
BALN (disk) technique	Control blocks
- being used SW1,56	
- on 2314 SW1,566	
· · · · · · · · · · · · · · · · · · ·	
- count of strings on	user library SORCE,788
each work area SEQCT,	
- alternate merge order MRGAI	
- optimum merge order MRGOP,	
<ul> <li>pointer to table of</li> </ul>	Control fields
formatted tracks FTTAB,	
BALN (tape) technique	field2 PCF02,120
- being used both SW1,560	
and SW1,56°	
- count of strings on	•
each work area SEQCT,	592 field12 PCF12,180
- current pass odd or	further fields
Blocking	to by ADDCF, 1048
- of input IPBLK,	
Buffer, packing	- rightmost column in-
- displacement of PBUFF,	
Buffers	- 'E' type present SW1,564.0
<ul> <li>size of input buffer LAB03,</li> </ul>	
- no. of input buffers LAB03,	.684 - 'D' type present SW1,563.2
- size of output buff. LAB07	.689 Control statements
- no of output buffs LAB07,6	
- alignment SW1,563	3.? - MERGE present SW1,564.2
- no of ph1 buffers BUF1,92	
- no of ph2&3 buffers BUF23,6	
- addr of input buffer	- RECORD absent or in-
- addr of ph1 buffer	Control word
pool LABO6,7	
- addr of merge-only	Counter (blocks)
buffer pool LAB06,7	
Channels	Counter (records)
<ul><li>types and combina-</li></ul>	<ul> <li>total records entering</li> </ul>
tions of channels SW1,565	3.3+4 the phase RCDCT,588
- tape switch (or al-	- records inserted INSCT,584
ternative channel	- records skipped SKPRD,80
option) present SW1,564	
- >1 channel available SW1,56	
Checkpoint	work units (ph2) COUNT,576
- being used SW1,562	
Communication area	- records deblocked fr
for user USER,67	
Comparison techniques:	Counter (strings)
- multiple techn used SW1,562	
	work area (BALN only) SEQCT,592

and the last		-9	0%3 E74 A
CRCX technique	094 ECE 0	- given exactly	SW2,571.0
- being used	SW1,565.?	- value of	FILSZ,604
- merge pass to follo	W 5W1,506.?	Final merge (phase 3) - size of work area	P3GC,700
Data chaining		- no. of bins (var)	BDSVA,745
- none	SW1,561.3	- addr of next bin (v	
- input	SW1,561.4	- addr of input buffe	al) bubva;;;
- output	SW1,561.5	table	BDSVA,744.1-3
DCB table	DH 1730 123	- addr of move list(f	
- address of	STDCB,760	- return addr to RCV	
Descending	DIDCD / 100	- pointers to modules	
- order for SORTOUT	SW1.563.2	RPG CO MOGULES	WRT,840
- order for merge pas		read	RD,856
Disk sort	5 611 1, 00 C C C	open list	OPEN,944
- addr of disk direct	ory DIRAD 280	deblock	DEB,816
- no of SORTWK areas		merge	NET,824
- no of blocks/track		block	BLK,832
- no of tracks/cyl.	TPCYL, 1068	messages (assignmen	
- no of directory blo		messages (running)	
per track	DPTRK, 1070	Fixed-length records (	
<ul><li>no of bytes/track</li></ul>	CPTRK, 1072	- being used	SW1,560.0
- RPS feature used fo		Generated core, perman	
SORTWK	SW1,565.?	- size of	PGCSZ,96
- device code for uni		- address of	ARGC, 1074
with RPS	DEVCD, 1106	<ul> <li>address of network</li> </ul>	table
- disk addresses,ph1,	2,3 DSKED,152	in area	NETAR, 1080
<ul> <li>disk starting addr</li> </ul>	STAR , 288	GETMAIN	
End of file		- addr of table of ad	dr GETMN,780
<ul> <li>reached for OSCL</li> </ul>	SW1,563.7	<ul> <li>addr of table of si</li> </ul>	zes GETSZ,784
Exits		Initiation of the prog	ram
<ul><li>none activated</li></ul>	SW1,561.6	- how achieved	SW1,563.5
<ul> <li>some activated</li> </ul>	SW1,561.7	Input buffers	
<ul> <li>those activated</li> </ul>	MODEX,568	- no & size of buffer	s LAB03,684
<ul> <li>pointers to routine</li> </ul>		<ul> <li>addr of buffer tabl</li> </ul>	e LAB02,724
for E11	X11,952	Input DCB	
for E21 or E31	Xx1,960	- size of SORTIN DCB	DCBIN, 1064
for E15	X15,968	Input device	
for E25 or E35	Xx5,976	- used as work device	SW1,565.5
for E17	X17,984	Input record blocking	
for E27 or E37	Xx7,992	- blocking factor (fi	
for E18	X18,1000	- blocksize (var)	IPBLK,88
for E28 or E38	Xx8,1008	Input record length	0714 ECO O
for E19	X19,1016	- up to 256 bytes	SW1,562.?
for E29 or E39	Xx9,1024	- >256 bytes	SW1,562.?
for E61	X61,1032	- LENGTH parameters f	
<pre>for E16 - linkage editor stat</pre>	X16,1040	SORT or MERGE state	RCDL1,648
- DDname of user mod		12	RCDL2,650
or DCB addr of SYSL		13	RCDL3,652
user lib	SORCE,788	14	RCDL4,654
- when program invoke		15	RCDL5,656
ph1 exit address	ATP1E,84	Input to a merge	1.0520 7 000
ph3 exit address	ATP3E,708	- present	SW1,563.7
Extract		- sequence for	SW1,563.3
- routine being used	SW1,560.2	Intermediate merge (p)	_
- length of control w		- addr of phase work	
- size of area used h		area	PDWA,72
extract routine	PGCSZ,96	- size of work area	P2GC,696
F field (constant used		- addr of input buffe	
- displacement of	FFF,100	table BDS	SVA,744.1-3
File size		- addr of move list(	
<ul><li>estimated</li></ul>	SW1,563.6	- no of avail. bins (	

		_	
<ul> <li>size and offset of</li> </ul>		- addr of conversion	
major control field		hex to char.	CONV,896
- return addr to RCV		Multiple compare routing	ne
<ul> <li>pointers to modules</li> </ul>	used:	<ul> <li>being used either</li> </ul>	SW1,560.3
algorithm	ALG,808	or	SW1,563.0
read	RD,856	<ul><li>not being used</li></ul>	SW1,560.2
deblock	DEB2,864	Odometer table	-
merge	NETM,872	(for OSCL)	ODOM,288
block/deblock	BLK2,880	Options	02011,220
write	WRT,840	- DIAG used	SW1,565.?
	OPEN,994	<del></del>	
open list		- FLAG (or MSG) option	11 F3K34,104
messages (assignmen	C) Wide 1330		
messages (running)	RMB,923	OSCL technique	0114 ECO C
Invoked		- being used	SW1,560.6
program invoked	SW1,563.5	- addr of unit count	
Job name	JOBNM, 1088	table	UNTCT,772
Linkage editing		<ul> <li>addr of input buffer</li> </ul>	
<ul><li>edited via sort/mer</li></ul>		table, ph2	LAB10,776
program: set for ea	ch	- pointer to module w	hich
exit in	LINK,572	initializes tree &	
Main storage		sorts	INT,888
- addr of next availa	ble	- RMAX (max bytes on	-
byte	SPGN 1,720	one work file)	XCAP,616
- addr of GETMAIN tab		- RMAX reached	SW1,566.?
of addresses	GETMN,780	- count of records de-	
- addr of GETMAIN tab		blocked from SORTIN	
of sizes	GETSZ,784		SW1,562.?
	-	- EOF	
- addr of permanently	3DGG 1078	- user EOF	SW1,566.?
generated area	-	- user insert in proc	ess Swi, Doo.?
<ul> <li>addr of network tab</li> </ul>		Output blocking	
in perm. gen. area	NETAR, 1080	- blocking factor (fi	
Merge		<ul><li>blocksize (var)</li></ul>	OPBLK,626
(see Final merge and		Output buffers	
Intermediate merge)		- no. & size of buffe	rs LAB07,688
Merge-only		<ul> <li>address of buffer 1</li> </ul>	LAB04,728
<ul><li>being used</li></ul>	SW1,561.?	- address of buffer 2	LAB05,732
- MERGE stmt present	SW1,563.?	Output DCB	
- assignment EOF		- size of SORTOUT DCB	DCBOU, 1066
- pointer to EODAD rt		Output device	•
(see also Final merge		- output unit for ph3	OPFMP
Merge order	•	- unit address	DEPHO,640
- maximum	MRGMX,658	Packing buffer	<b>5</b> 22 <b>,</b> 5
- alternate (BALN)	MRGAL,660	- displacement of	PBUFF, 104
- optimum (disk sort)	MPGOP 662	Phase	12011/104
- · ·	MIGOL 7002		SW1,560
Messages		- which in progress	
- print option used	D2500 700	ph1	SW1,562.3
(FLAG or MSG param)	P3A5Z, 104	ph2	SW1,562.4
- length of VSAM msg		ph3	SW1,562.5
area	VSMSL,1118	Phase 1	
- pointers to msg mod		(see Sort)	
ph1 assignment	AMA,928	Phase 2	
ph1 running	RMA,912	(see Intermediate mer	ge)
ph2 assignment	AMB,936	Phase 3	
ph2 running	RMB,932	(see Final merge or M	erge-only)
ph3 assignment	AMC,936	Phase work area	- <del>-</del> -
ph3 running	RMC,932	•	
VSAM	VSMSG, 1120		
	· · · · · · ·		

			242 040
Read/Write		messages, running	RMA,912
- directory	LAB01,200	open list	OPEN,952
- error flags	SW1,565.0-1	EODAD routine	EOF,904
<ul><li>block/deblock save</li></ul>		Sorting	
area	BDSVA,744	<ul> <li>pointer to program</li> </ul>	
<ul> <li>addr of DCB table</li> </ul>	STDCB,760	work area	WKARE,72
<ul> <li>addr of IOB table</li> </ul>	STIOB,768	- device types allowe	d DEV,576
Record length	•	- G (capacity of RSA,	
(see Input record len	ath)	in records)	PPISRTG,620
RPS	, <b>,</b> ,,	- B (work storage	
- used	SW1,564.3	blocking)	SRTBL,624
RSA	5111,504.5	- capacity of work	SKIDH JUZT
	_	<del>-</del>	VOND 646
- capacity, in record		areas	XCAP,616
(G)	SRTG,620	Spanned records	
- no of entries in th		- used in input	MODEX,570.6
<ul> <li>addr of table</li> </ul>	LAB08,741	<ul><li>used in output</li></ul>	MODEX,570.7
- bin size	BINSZ,608	Step name	STPNM, 1096
<ul> <li>pointer to next</li> </ul>		Tape sort (general)	
available bin	???	<ul> <li>tape table pointer</li> </ul>	TPPT,244
Sequence check		- tape table	TPTBL,248
- user option for	DOUO,692	- inter-record gap	·
- addr of record	DOOBA,740	size, in bytes	IRG,612
- addr of record in o	ntnut	- some or all tapes a	
buffer in ph2	OPBAD, 1108	7-track	SW1.567.4
SKIPREC	OF BAD, 1100		
	0M4 E60 0	- tape switch (or alt	61- 11 CM1 E6E 6
- being used	SW1,562.2	native channel) ava	
- count of records		- deblock backward	SW1,566.0
skipped	SKPRD,80	- read forward	SW1,566.1
Sort (phase 1)		<ul> <li>close with rewind</li> </ul>	SW1,566.2
- assignment size	P1ASZ,76	<ul> <li>block forward</li> </ul>	SW1,566.3
<ul><li>no of buffers</li></ul>	BUF1,92	<ul> <li>read forward later</li> </ul>	
<ul> <li>size of phase work</li> </ul>		Techniques, sequence d	istribution
area	P1GC,72	- BALN (tape) both	SW1,560.4
<ul> <li>size of available</li> </ul>		and	SW1,561.2
main storage	TAVLC,712	- POLY	SW1,560.5
- addr of input buffe		- OSCL	SW1,560.6
pool	LAB06,736	- BALN (disk)	SW1,560.4
- no of avail bins (va			SW1,560.2
		- CRCX	SW1,566.6
- addr of next bin (va	TI DUSVA,148	Tree	D4 1 1 200 00
- entry to move rtn	DD 0713 757		mmm 716
(var)	BDSVA,757	- end address of	TREND,716
- addr of move list (f		- addr of initializin	
- collating order	SW1,562.?	module (OSCL)	INT,888
- return addr to RCV		Variable-length record	
- pointers to modules		- present	SW1,560.1
algorithm	ALG,808	VSAM	
read	???	<ul> <li>addr of VSAM read</li> </ul>	
VSAM read	VSI,1124	module, ph1	VSI,1124
deblock	DEB, 816	- addr of VSAM msg ar	
sort	NET,824	Work areas, pointers t	
block	BLK,832	- program area (sort)	
write	WRT,840	- program area (merge	
move (var)	VMV,848	brodram area (merde	, EDNATIE
messages, assignmen			
messages, assignmen	C AMA, 720		

# Module ICEAMI: Generated Defaults

The option values generated when the program is installed are stored in module ICEAM1. The format of the module is shown below. Values underlined are generated by default if no specification is made when the program is installed. The generated values can be changed by use of the IMASPZAP, HMASPZAP (VS1) or AMASPZAP (VS2) Service Aid Program.

	DISPI DEC	ACEMENT (HEX)	SIZE bytes	FIELD NAME	CONTENTS
	0	0	4	SIZE	X'00FFFFF8' (representing MAX) or size value in hexadecimal
	4	4	4	MAXLIM	Representing MAXLIM value in hexadecimal (Default: X*00080000*)
1	8	8	4	MINLIM	MINLIM value in hexadecimal (Default: X'00012000')
	12	c	4	RESALL	All disk sorts reserved main storage value in hexadecimal (Default: <u>X'00001000')</u>
	16	10	4	RESINV	Invoked sorts reserved main storage value in hexadecimal (Default: <u>0</u> )
	20	14	4 <u>1</u>	MSGS	'NYYN' (FLAG (I) ) 'NYNY' (FLAG (U) ) 'YNNY' (NOFLAG) 'YNYN' (AC)
	24	18	1	EQUALS	'Y' (YES) ' <u>N</u> ' (NO)
	25	19	1	LIST	'Y' (YES) 'N' (NO)
]	26	1A	1	ERETJCL	'Y' (ABEND) ' <u>N</u> ' (RC16)
1	27	1B	1	ERETI NV	'Y' (ABEND) ' <u>N</u> ' (RC16)
1	28	1C	8	PRINT	Message DDname for invoked sort (Default: <u>"SYSOUT"</u> )

Licensed Material--Property of IBM
Page of LY33-8042-6, As Updated 31 March 1981, By TNL LN20-9345

1	DISPLA DEC	CEMENT (HEX)	SIZE bytes	FIELD NAME	CONTENTS
	36	24	1	VIO	'Y' (YES)
į	37	25	1	EXCPVR	'Y' (YES) 'N' (YES)
İ	38	26	1	RESDNT	'Y' (ALL) 'M' (MON) 'N' (NONE)
İ	39	27	1	RELEASE	' <u>Y</u> ' (YES) 'N' (NO)
[	40	28	1	SECALL	'Y' (YES)
İ	41	29	1	VERIFY	'Y' (YES) ' <u>N</u> ' (NO)
į	42	2A	1	BLKSET	' <u>Y</u> ' (YES) 'N' (NO)
1	43	2В	1	CHALT	' <u>C'</u> (YES) 'A' (NO)
	44	2C	2	svc	X'0A6D' (the operations code for SVC 109 or SVC 200 through 255 in hexadecimal)
	46	2E	1	СНЕСК	'Y' (YES) 'N' (NO)
	47	· 2F	1	SMF	'F' (FULL) 'S' (SHORT) ' <u>N</u> ' (NO)
	48	30	1	VBLKSET	' <u>Y</u> ' (YES) 'N' (NO)
1	49	31	256	ALTSEQ	Translate table for AQ control fields (Default: standard <u>EBCDIC</u> sequence)

Note: If you have changed the generated values and reenterable modules are placed resident, you have to re-IPL the operating system to activate the changes. For SVS/MVS specify option CLPA when you re-IPL.

# Section 6. Debugging Aids

| This section is intended to help you if Sort behaves in an unexpected way, and you want to localize the problem and if possible solve or bypass it. It contains the following subsections:

- How to decide whether a problem is due to a program error.
- General considerations when an error has been located.
- How the microfiche are organized.
- Adding a temporary change to the maintenance area.
- How Sort uses registers.

1

- How to read a dump to find DCBs and IOBs for SORTIN, SORTOUT, and SORTWK.
- Various uses of the DEBUG control statement.
- Explanations of the diagnostic messages.
- Explanation of program dumps.
- Finding an object module in a storage dump.
- Message-to-module cross reference. How to find messages in the code.
- Cross references showing the displacement of each field in the common data areas (COMMON, COMMA, CPI, and PPI) for the various techniques.
  - Other diagnostic aids are provided in Section 3, where the phase structures are given; Section 5, which shows the layout of some important control blocks; and Appendix A, showing which modules activate the various program exits.

# ! Defining Problem Cause

| If the program is unable to successfully complete sorting or merging, you will certainly get one or more program messages, and possibly also | an ABEND code.

| The OS/VS Sort/Merge Programmer's Guide gives explanations of the | various program messages, and suggestions as to how to cope with them. It is assumed that you have exhausted those explanations before turning | to this section.

#### | IS THIS A PROGRAM ERROR?

| Your first task is to decide whether or not the problem is due to an | error in Sort code.

If your installation has just installed a new release or PTF level of Sort, it is worth checking that any necessary additional alias names have been added to module ICEMAN. If they have not, mixed levels of program modules can be executed, which can give rise to unpredictable abnormal terminations.

Otherwise, if Sort is run alone in its region, problems are unlikely to a rise from the environment. If no routines of yours were invoking Sort, or being used at program exits, you can therefore work on the assumption that you have found a program error, and turn to the section 'Bypassing the Problem' below.

| However, if you are invoking Sort from a program of your own, or if you | are using routines at program exits, you will need to eliminate your own | programs as sources of error. In the example in Figure 16, for | instance, an exit (E15) is being used.

l l	ICE0001	CONTROL STATEMENTS/MESSAGES 5740-SM1 REL 4.0
	ICE0741 ICE0881 ICE0931 ICE039A	SORT FIELDS=(1,5,CH,A),EQUALS RECORD TYPE=F,LENGTH=(1200,,1000) MODS E15=(E15,79000,MODSLIB,N) RECORD LENGTH L1 OR L3 OVERRIDDEN SORTJOB.SORTSTEP, INPUT LRECL=1200, BLKSIZE=12000, TYPE=F MAIN STORAGE = (MAX,262144,48528), NMAX=7300, PEERAGE INSUFFICIENT MAIN STORAGE - ADD 6K BYTES

| Figure 16. A Sample Set of Messages

#### | POTENTIAL PROBLEMS WITH ROUTINES AT PROGRAM EXITS

## | Use of Registers

The first thing to check with your routines is that they observe the standard linkage conventions. If, for example, they corrupt register 12, results are unpredictable but almost certain to result in an ABEND of some kind, because Sort uses register 12 as its main base register.

| Check, too, that you are not using registers for loading or storing that | are accidentally causing overlay of sort code or work areas. If this | happens Sort could work without errors with one technique, but fail with | another.

#### Space

The next thing to check is whether your routines are trying to use more space than you have allocated to them. Have you installed a new operating system release since the last time you used these routines? Each time you use an OPEN macro, for example, your program may take buffer space; but the amount it tries to take will depend upon such factors as the current release of the operating system.

A change of operating system could therefore lead to an ABEND in your own routine; or it could lead to too little space being left for Sort.

You can see whether too little space was left for sorting by studying the information in message ICE093I. As shown in Figure 16, the third field in the 'MAIN STORAGE' parameter will tell you how much was actually left for Sort after your own routines had taken what they needed, in a region or partition of only 256K. Since in the example the input block size is large, the 48,528 bytes left are not enough.

Similar situations can occur if Sort is dynamically invoked using the MAX option, and a fairly large reserved value is passed to Sort or taken by default.

Another problem could arise if the E15 routine issues a GETMAIN without a corresponding FREEMAIN at the end. This problem can be caused indirectly, for example by leaving a data set open so that a buffer pool remains reserved.

#### Record Contents

If the output records do not appear to contain the same data as the input records, and either E15 or E35 has been used, check that your routine is handling register 1 correctly; especially, check that it is correct on return to Sort.

If for example you first load register 1 and then restore all registers (including register 1), it will probably have the wrong contents.

Equally, if you first restore all registers and then try to load register 1 from a corrupted base register, you will almost certainly pass the wrong information to Sort.

# POTENTIAL PROBLEMS WITH INVOKING PROGRAMS

Space can also be a problem when you invoke Sort from another program, especially if you are using SIZE=MAX and invoking exit E15 or E35 (or, from COBOL, using an Input or Output procedure).

If you do this, and particularly if you open a file in your exit routine, check that you specify a sufficiently large amount of reserved storage.

Licensed Material--Property of IBM
Page of LY33-8042-6, As Updated 31 March 1981, By TNL LN20-9345

## Considerations when an Error has been Located

When you have pinpointed an error to a particular module, it is always worth doing a RETAIN search to find out whether the problem is already known. If the problem itself has not been reported, you can search for other APAR fixes to the failing module, and apply them if appropriate.

If this procedure fails, you have the choice between trying to fix the problem, and trying to bypass it. You will then also want to report the problem.

# WHEN NOT TO WASTE TIME ON REPAIR

Some modules contain extremely complex code. Attempts at field repair can be expected to take an unreasonable amount of time.

The following load modules are particularly unsuitable candidates for field repair:

1	ICEIPUT	ICEIPVT	ICEOPUT	ICEOPVT
	ICEKPUT	ICEKPVT	ICEPAR	
1	ICEKPUS	ICEKPVS	ICERED	

# BYPASSING A PROBLEM

The simplest way of bypassing a problem in the program is to force it to use a different technique. Message ICE092I or ICE093I will tell you which sorting technique has been used.

You can use the DEBUG control statement, described later in this section, to force the use or non-use of a specific technique.

Alternatively, if the problem is in the Blockset technique (the newest in the program), you can 'turn off' that technique altogether by including an OPTION statement with the parameter NOBLKSET.

#### | REPORTING A PROBLEM

I You have two alternative means available for reporting a problem: an incident report, or an APAR.

## | Submitting an Incident Report

- Generally, central service needs the following information in a RETAIN incident entry:
- Symptom
- Was SORTIN or SORTOUT DD statement present? If so, list the DCB attributes.
- Was Sort invoked from another program? If so, try to list the contents of the parameter list passed. See the OS/VS Sort/Merge Programmer's Guide for parameter list format. At entry to Sort register 1 must point to the parameter list. Note, however, that (except with a standard disk sort) the initial contents of register 1 are lost towards the end of Phase 0.
- Was Sort invoked via JCL? If so, list all the program control statements used.
- List Release and PTF levels, and if possible the APAR fixes applied since the last PTF.
- Give the number of SORTWK data sets used, and their unit type.
- If message ICE092I or ICE093I was issued, give its contents.
- If you have a dump, list the Sort module (s) in storage when the program failed, and the displacement into the failing module at which the abend occurred.

#### ! Submitting an APAR

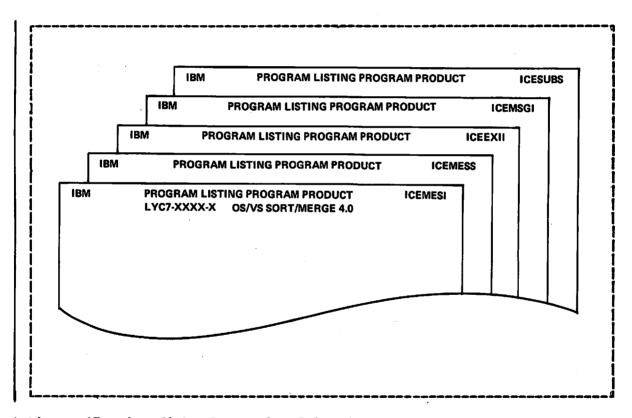
- | In addition to the information requested on an APAR form, you should supply the following information:
- A dump from the failing run of Sort.
- A list of PTFs applied, and if possible the APAR fixes applied since the last PTF.
- Source code (Assembler or compiler listings) and object code for any user routines and/or invoking program.
- A listing of the JCL.
- A copy of the input file (s) in machine-readable form.
- 1 A list of program control statements used.

 A list of the generation options selected, which can be produced by using IMASPZAP (or an equivalent program) to dump the contents of module ICEAM1.

# | Microfiche Organization

| For Peerage and Vale each load module is on one fiche, with the same | name as the load module name. For example, module ICEDEF is completely | contained on a fiche called ICEDEF.

| Blockset has several fiches for each load module. Each object module is on a separate fiche. For example, load module ICEMESI consists of five object modules (ICEMESI, ICEMESS, ICEEXII, ICEMSGI, and ICESUBS), and is on five fiches with those names, as shown in Figure 17.



| Figure 17. Microfiche for Load Module ICEMESI

| The other techniques have one fiche per load module, except for modules | ICERCM and ICERCZ. These two modules have an overlay structure, and are | shown with one fiche per object module. Section 3 gives further details | of module organization under the heading 'Phase Structures'.

# Adding a Temporary Change to the Maintenance Area

| Every new Sort module has a maintenance patch area.

| For Peerage and Vale the area is included in each load module. It is 1 100 bytes long, or 5% of the size of the module, and is at the end of the module. It is usually addressable by using register 11 plus the appropriate displacement (which can be calculated from the microfiche), and generally starts with the label PATCH in the source code.

| Blockset has a maintenance patch area of at least 64 bytes in each object module. It can be found in the fiche by looking for the label | PATCH.

The other, older techniques do not have patch areas. Instead you can use the EXPAND facility of the linkage editor to create one at the end of a module, if required.

| To use a patch area, first verify that it contains all zeros. the appropriate utility (IMASPZAP, HMASPZAP, or AMASPZAP) to put in the | required change.

| If the Sort reenterable modules are in the LPA library, do not forget to re-IPL with the CLPA option.

# | How Sort Uses Registers

| In addition to the conventions described in Appendix D, the Peerage/Vale techniques make extensive use of certain internal register conventions. 1 They are shown in Figure 18.

	Register	Use
1	0 - 5	Work registers
1	6	Current RSA pointer
1	7	Pointer to current record in input buffer for SORTIN or SORTWK
	8	Pointer to current record in output buffer for SORTWK or SORTOUT
	10	Internal return register when branch-and-link to sub-routines is used
1	11	Secondary base register
ı	12	Primary base register
1	13	Pointer to the common area (COMMA or COMMON)
	14 - 15	Work registers

| Figure 18. Use of Registers by Peerage/Vale Techniques

# Finding DCBs and IOBs for SORTIN, SORTOUT, SORTWK

I This subsection describes how to locate the I/O control blocks in a system dump, for the standard disk techniques.

| The first step is to verify, from message ICE092I or ICE093I, which of the standard techniques has been used.

| You should then find the common area. Look at the address pointed to by register 13, and check whether the first word contains the characters 'SM1'. If it does, register 13 is intact, and can be used when calculating offsets. If it does not, and you are certain the Sort program is being used, then register 13 has been corrupted. However, you can still find the common area by looking for the eyecatcher 'SM1' with which it begins.

# BLOCKSET TECHNIQUE

#### | SORTIN and SORTOUT Control Blocks

- | Figure 19 shows an example of interpreting a system dump for Blockset.
- | The procedure for finding input and output control blocks is as follows:
  - 1. Look in COMMON at label COMINDCB, which is at register 13 + X'154'.

    There you will find a pointer to the SORTIN DCB.
  - 2. The SORTOUT DCB pointer is at label COMUTDCB, at register 13 + X'158'.
  - 3. There is no pointer from the DCBs to the corresponding IOBs, because EXCP is used for access. However the IOBs are in the same areas as the corresponding DCBs, at displacement X'48' from the start of the DCB.
- 4. A fullword at IOB + X'14' points back to the DCB.

## | SORTWK Control Blocks

| The procedure for finding SORTWK control blocks is as follows:

- 1. Look in COMMON at label COMWKDCB, at register 13 + X°15C'. This is a pointer to the DCB list.
- 2. Follow the pointer to the SORTWK DCB list. Each element in the list is a pointer to a DCB. The last element in the list has the first bit on.
- 3. As for SORTIN and SORTOUT, the access method used is EXCP. The IOB for each work area follows the corresponding DCB, at DCB start address + X'48'.

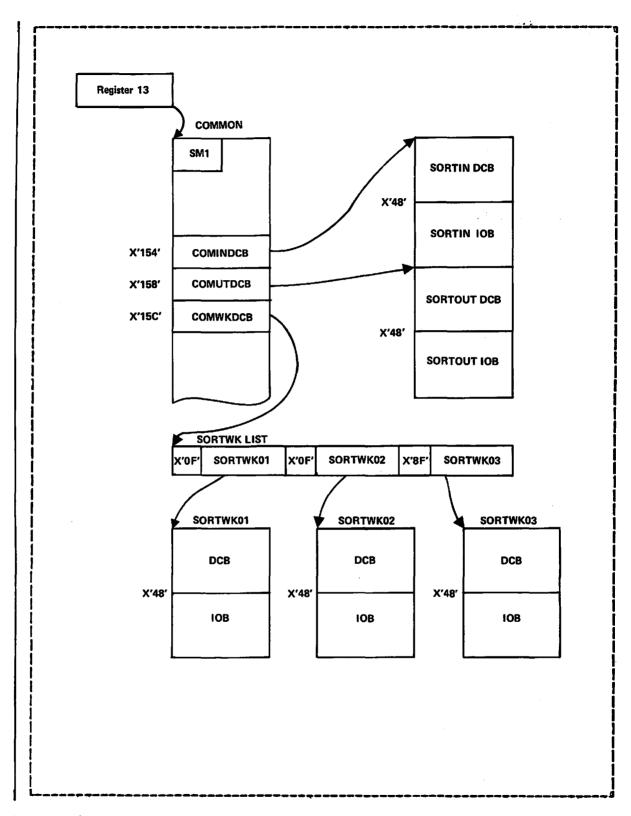


Figure 19. Locating Control Blocks in a Dump with Blockset

## | PEERAGE AND VALE TECHNIQUES

| The layout of all Peerage and Vale internal work I/O control blocks can | be found in the microfiche for modules ICEMON and ICEDEV.

## | SORTIN and SORTOUT Control Blocks

| First check which access method has been used. If EXCP was used, you | will have received message ICE084I; otherwise (if SAM was used) you will not.

- Field CSORTIN in COMMA, at register 13 \* X°5FC°, points to the SORTIN DCB.
- 2. Field CSORTOUT, at register 13 + X 600', points to the SORTOUT DCB.
- 3. If <u>EXCP</u> is used, most I/O information is in a storage area, subpool 0, with label OBSBLOCK. See Figure 20. A pointer to this area is in ICECOMMA at label COBSBLOK, at register 13 + X°7E4°.
  - At OBSBLOK \* X'8' there is an eyecatcher consisting of the characters 'OBSBLOCK'.
  - The layout of OBSBLOCK is given in the microfiche ICEOBS and ICEXCP.
  - a. In one of them, find the offsets to label OBSIOBP.
  - b. Add the value of the EQU' (OBSIGEP) to the address of OBSBLOCK (taken from register 13 + X'7E4', as described above). This gives the address of the IOB: if you are in Phase 1 it will be the SORTIN IOB; in Phase 3 it will be the SORTOUT IOB. If you are not sure of the phase, check the DCBs to see which one is open.

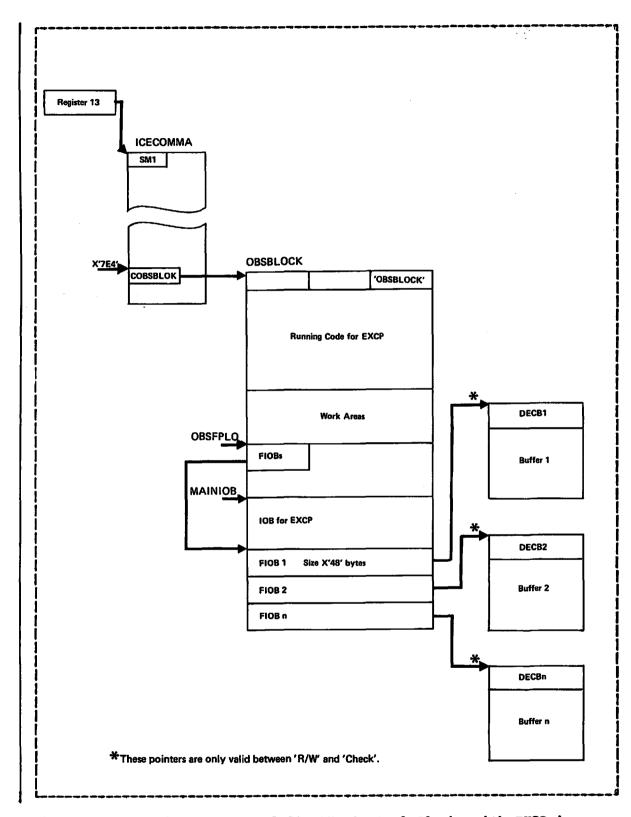


Figure 20. Locating SORTIN and SORTOUT Control Blocks with EXCP in a Dump with Peerage and Vale

- 4. OBSBLOCK also contains read/write and check simulators, used to simulate SAM access methods. The SORTIN and SORTOUT DCBs contain the addresses of the simulators, at DCB + X'30' and DCB + X'34' respectively.
- 5. After a read or write macro has been used to invoke a simulator, and before the subsequent check macro is issued, there are also valid pointers from OBSBLOCK to the DECB at the beginning of each buffer. See Figure 20. The address of the pointer list (a list of fake IOBs, or FIOBs) is at label OBSIFFIO. To find a DECB:
  - a. Look in the fiche for ICEOBS or ICEXCP, and find label OBSFIOBP.
  - b. Add the value of the EQU (OBSFIORP) to the address of OBSBLOCK, taken from register 13 + X\*7E4\*.
  - c. The result is the address of a pointer to the first FIOB.
  - d. The other FIOBs follow consecutively, each X°48° bytes long on a doubleword boundary.
  - e. The DCB NCP value tells how many valid FIOBs there are.
- f. The FIOB plus 4 points to the DECB.

Í

1

## SORTWK Control Blocks

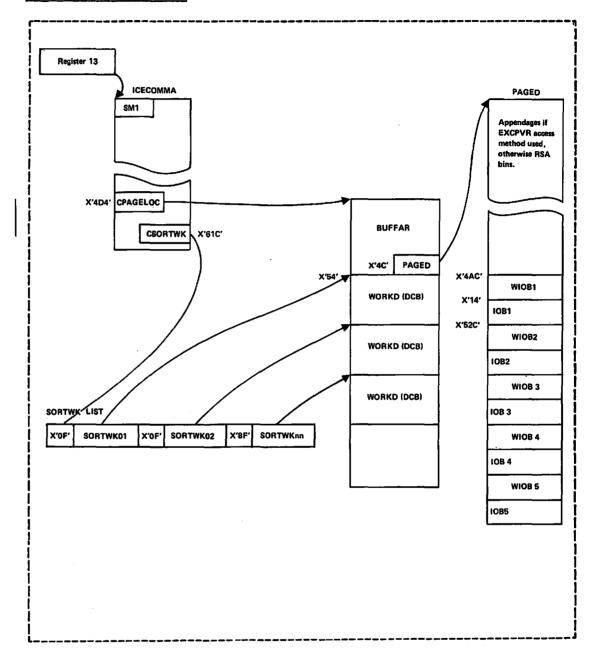


Figure 21. Locating SORTWK Control Blocks in a Dump with Peerage/Vale

Licensed Material--Property of IBM
Page of LY33-8042-6, As Updated 31 March 1981, By TNL LN20-9345

The procedure to find a work IOB is as follows:

- Look in COMMA at label CPAGELOC, at register 13 + X'4D4'. This field contains a pointer to a control block called BUFFAR.
  - Follow the pointer to BUFFAR. At BUFFAR + X'4C' you will find a pointer to a control block called PAGED.
  - Follow the pointer to PAGED. At PAGED + X'4AC' is an area, WIOB, containing five control blocks.
- 4. WIOB + X'14' contains the IOB. IOB + X'14' points to the DCB.
- The WIOB control blocks are each X'80' bytes long, and follow each other consecutively.

You can also go straight to the DCB, as follows:

I The field CSORTWK in COMMA, at register 13 + X'61C', contains a pointer to the SORTWK DCB list. Each word in the list points to a work DCB, which forms part of a control block called WORKD. The last word in the list has the first bit on.

### The DEBUG Control Statement

This statement is only valid when the program meets the criteria for the standard disk techniques. If it is supplied under other circumstances it is ignored.

The statement is not intended for regular use; only the first two parameters are of general interest. The other parameters can be used to provide a temporary bypass, or to supply detailed information on program execution for use when optimizing or debugging the standard disk sort.

DEBUG can be passed to an invoked sort by means of the SORTCNIL DD statement, for example:

//SORTCNTL DD \*
DEBUG BSAM

This DD statement is described in full in the <u>OS/VS Sort/Merge</u>
<u>Programmer's Guide</u>. Note that the DD name might not always be SORTCNTL,
because the first four letters of Sort's special DD statement names can
be changed for an invoked application. It might, for example, need to
be called //TESTCNTL instead. See the chapter in the Programmer's Guide
on invoking Sort from another program. The Blockset techniques will
not be used when the DEBUG control statement is used with a SORTCNTL
DD statement.

#### SYNTAX

(label) DEBUG ABEND ,DUMP [,PEERVALE] ,BALN
NOABEND ,NODUMP ,CRCX
[,CLOCK] [,FLAG] [,CTRX] [,BSAM]

ABEND NOABEND Overrides the generated default for action to be taken when the program encounters an uncorrectable error.

DUMP and NODUMP

Recognized but ignored.

**PEERVALE** 

With a disk sort one of the three standard techniques (Blockset, Peerage, or Vale) is normally used. If you have encountered a problem when using Blockset (see message ICE092I or ICE093I), you can temporarily bypass this technique by specifying PEERVALE, thus forcing Peerage or Vale to be used.

**BSAM** 

With the disk sort techniques Peerage and Vale, normally the EXCP access method is used for SORTIN and SORTOUT. If you encounter a problem related to this I/O activity you can temporarily bypass it by specifying BSAM.

BALN CRCX With a disk sort you can use this parameter to force either the balanced (BALN) or crisscross (CRCX) disk sorting technique and therefore bypass the standard disk sort techniques used by the program. If either BALN or CRCX is forced then the following restrictions apply:

- At least three work data sets on the same type of device are needed, with amount as specified in OS/VS Sort/Merge Programmer's Guide. Mixed device types are not allowed.
- Maximum record length must be less than work device track length.
- Allocation must be contiguous (the CONTIG parameter is required), and only primary extents will be used.
- Six or more work data sets are required for the CRCX technique.
- For SORTWKnn: nn can be any number from 01 to 32.
   The first number <u>must</u> be 01 and the others must follow consecutively with no gaps.
- Unused work space will not be released; the RLSE parameter must not be specified.

1 1	CLOCK (only Peerage and Vale)		structs the program to measure elapsed and CPU mes for the different phases.							
1 1	FLAG (0) (only Peerage and Vale)	(I	structs the program to print information messages CE120-125). These messages are listed under essages Produced by Using the DEBUG Statement.							
1   1   1   1	CTRx=value	to in va	Specifying this parameter will force Peerage or Vale to be used. The program will keep a count of the input or output records. When the count reaches the value specified the program will ABEND, and a formatted dump will be printed.							
1		Th	e numbers that may be assigned to x are:							
1		2	Count of input records being moved from the input buffer.							
1		3	Count of output records being moved to the output buffer.							
1		4	Count of input records inserted by E15.							
ı		5	Count of output records deleted by E35.							

Note: When the count reaches 'value', the program will ABEND. It will also terminate with message ICE025A if the 'value' is a number greater than the number of input records.

## | MESSAGES PRODUCED BY USING THE DEBUG CONTROL STATEMENT | Messages ICE120-125 are issued if the DEBUG statement is supplied with the appropriate parameters FLAG(@) (only for Peerage and Vale sorts). | ICE1200 RL=a B=b IL=c IS=d IB=e RM=f EM=g BA=h IX=j OX=k Explanation: This message relates to the Optimization part of the definition phase (Phase 0). RL record length (within the Sort); blocking factor used for work areas; В IL number of physical index blocks per logical index block; IS index entry size; IB number of indexes/physical index block; RM maximum number of strings to be merged in one pass of Phase 2; EM maximum number of strings to be merged in Phase 3; BA base bin size; IX number of input buffers; OX number of final output buffers. | ICE121C ET=a CT=b BN=c X=d TO=e SN=f G=q Explanation: This message relates to Phase 1. ET elapsed time taken in centiseconds; CT CPU time in centiseconds; BN number of blocks handled; number of EXCPs issued; X TO number of tracks put out; SN number of strings produced; number of records in the record storage area. ICE122R ET=a CT=b BN=c X=d RM =e PN=f BT=g TO=h Explanation:

This message relates to Phase 2 (Reduction).

elapsed time taken in centiseconds; CT CPU time in centiseconds; BN number of work data set blocks handled; number of EXCPs issued; number of records in the record storage area; RM maximum number of strings to be merged in one pass of Phase 2; PN highest partition number; BT number of tracks handled more than once. TO number of tracks put out;

```
G
 ICE123E ET=a CT=b BN=c X=d
 EM =e TO=f BT=q
 Explanation:
 This message relates to Phase 3.
 elapsed time taken in centiseconds;
 CT CPU time in centiseconds;
 BN number of work data set blocks handled;
 number of EXCPs issued;
 number of records in the record storage area;
 EM maximum number of strings to be merged in
 Phase 3;
 TO number of tracks put out;
 BT number of tracks handled more than once.
I ICE124P ET=a CT=b PE=c RP=d CX=e CO=f C0=q CR=h G=i WB=i
 Explanation:
 This message relates to Phase 2 (Partitioning).
 ET elapsed time taken in centiseconds;
 CT CPU time in centiseconds;
 'peerage': the number of logical strings
 obtained by logically rearranging the tracks of physical strings;
 RP number of partitions;
CX number of exempt blocks;
CO number of overflow blocks;
 CO number of blocks in partition 0;
 CR number of blocks to be handled by partition 0;
 number of records in the record storage area;
 WB number of blocks written back to work storage.
I ICE1250 CT=a GP=b SA=e X=d
 Explanation:
 This message relates to work I/O, and is cumulative:
 it appears after each of Phases 1-3, and shows
 cumulative totals each time.
 CT CPU time in centiseconds;
 number of work I/O blocks; number of standalone seeks;
 GP
 SA
 number of EXCPs issued.
 X
```

## | Messages Produced by Using the DIAG Option

| Diagnostic messages are obtained when you specify the DIAG option in the | PARM field of the EXEC job control statement. This option is only | available for tape techniques, a merge-only application, or when forcing | a non-standard disk technique.

| The DIAG option impairs program performance, and should be removed as | soon as it is no longer needed.

! The diagnostic messages are as follows:

1	!	
1	ICE9001 GENERATED CORE END ADDRXX	ICE926I IOB TBL ADDR xxxx
ı	ICE901I INPUT BFR TBL ADDRXXXX	ICE927I I/P CCW ADDR xxxx
1	ICE902I OUTPUT BFR ADDR xxxx,xxxx	ICE940I GENERATED CORE END ADDR
i	ICE9031 RSA TBL ADDR XXXX	ICE941I INPUT BFR TBL ADDR xxxx
1	ICE9041 TREE ADR FROM XXXX TO XXXX	ICE942I OUTPUT BFR ADDR XXXX,XXXX
ł	ICE905I MOVE RTN ADDR XXXX	ICE943I MOVE RTN ADDR xxxx
1	ICE9061 DCB TBL ADDR XXXX	ICE944I ECB TBL ADDR xxxx
1	ICE907I O/P CCW ADDR xxxx	ICE945I I/P CCW ADDR xxxx
ı	ICE908I OUTPUT IOB ADDR xxxx	ICE961I TECHNIQUE xxxx
ı	ICE909I OPEN LIST ADDR xxxx	ICE962I NO/SIZE OF BFRS, PH x, x, xxxx
i	ICE920I GENERATED CORE END ADDR xxxx	ICE963I MAX.SYSGEN CORE XXXX
1	ICE921I INPUT BFR TBL ADDR XXXX	ICE964 CALC. CORE PH1=xxxx
ı	ICE922I OUTPUT BFR ADDR xxxx,xxxx	ICE965I MERGE ORDER=xxxx
١	ICE9231 MOVE RTN ADDR XXXX	ICE988I ICEyyy LOC. AT xxxx*
1	ICE924I DCB TBL ADDR	ICE989I CLOCK - xx,xx,xx²
1	ICE925I O/P CCW ADDR xxxx	ICE9901 NO OF STRINGS PROD BY PH1 XXXXXXX

<sup>4</sup>Appears frequently; provides the starting addresses of the program modules.

<sup>| 2</sup>Appears at the beginning of each phase (except Phase 0), and at the end of the program.

## Dumps

| There are two types of failure that can cause dumps:

- Sort-detected uncorrectable errors which give critical error messages.
- Sort program failures that are detected by the operating system.

#### NORMAL ABEND DUMPS

| The default ERETINV|ERETJCL=ABEND|RC16, which was set when Sort was installed, can be overridden in a standard disk sort by the DEBUG control statement or by the PARM Field Option DIAG. To obtain a normal ABEND dump you must provide a SYSUDUMP or SYSABEND DD statement.

An invoked tape sort program, upon detecting an error, will give a return code of 16 unless DIAG is specified in the PARM Field Option, and a SYSUDUMP or SYSABEND DD statement is provided.

#### THE TRACE TABLE

In main storage the TRACE table is stored for Blockset in COMMON at COMTRACE (Register 13 + X'818') and for Peerage and Vale in ICECOMMA at CTRACE (Register 13 + X'814'). In a formatted dump it is eliminated from ICECOMMA dump and shown separately at the end of the FORMATTED DATA (see Figure 22). The TRACE table can hold up to 128 trace records, each one byte long. The table is divided into four fields:

PIELD 1	FIELD 2	FIELD 3	FIELD 4			
4 bytes	32 bytes	32 bytes	60 bytes			

! The table is initialized with zeros.

#### How Events are Recorded

Four types of events are traced

Type	Blockset	Peerage and Vale					
Type 1 Type 2 Type 3 Type 4	Record level Block level Blockset level Mainly module level	Record level Block level Main loop level Module level					

The object is to keep a long trace of Type 4 events, a shorter trace of Types 3, and 2, and only a short trace of Type 1. For this reason record insertion is carried out as follows:

- A new event is inserted at the end of the field corresponding to the type number; Type 4 goes at the end of field 4, Type 3 at the end of field 3, etc.
- All records to the left of the new one move one byte to the left; the leftmost one (in Field 1) is thus lost.

### Format of an Event Record

The Event record is one byte long, and consists of a description code. There are 255 codes ( $X^{1}00^{\circ} - X^{\circ}FF^{\circ}$ ).

## Licensed Material--Property of IBM Page of LY33-8042-6, As Updated 31 March 1981, By TNL LN20-9345

## EVENT-TABLE DESCRIPTION FOR BLOCKSET

		DIOCUCET	
TRACE SEGMENT	MODULE	BLOCKSET	DESCRIPTION
1,01 1,02 1,02 1,03 C4COBIN 1,03 1,04 C4COBIN 1,04 1,07 C4LOAF35 1,08 C4LOAF35 1,09 C5REXE35 1,00 C5REXE35 1,00 C5REXE35	ICECOBV ICEE15B ICECOBV ICEE15B ICEIFUM ICEIFUM ICEIFUM ICEE35B ICEE35B ICEE35B ICEE35VL ICE35VL ICE35VL		E15 CALL, COBOL MODE CALL E15 E15 ANSWER, COBOL MODE RETURN FROM E15 CALL E15 E15 CALL, SORTIN MODE RETURN FROM E15 E15 ANSWER, SORTIN MODE CALL E35 RETURN FROM E35 CALL E35 E35 CALL, SORTOUT MODE RETURN FROM E35 E35 CALL, SORTOUT MODE RETURN FROM E35 E35 ANSWER, SORTOUT MODE E35 ANSWER, COBOL MODE E35 ANSWER, COBOL MODE
2,11 C4DXLBLD 2,12 C4CHKIO 2,13 C4PUTVIR	ICEIPUB, ICEIPUB, ICEKPUV	KPUB, OPUB KPUB, OPUB BLOCKSET LE	DATA TRANSFER LIST BUILD CHECK I/O TRANSFER PUT VIRTUIAL BLOCK RECORD VEL
3,21 C4PICKUT 3,22 C4MAKKAD 3,23 C4CONVIM 3,24 C4CONVIM 3,26 C4PARBLK 3,27 C4GETSPC 3,28 C4KYBVDK 3,29 C4KEYIO 3,28 C4COPOUT 3,28 C4COPOUT 3,28 C4COPOUT 3,28 C4COPOUT 3,28 C4COPOUT 3,28 C4COPOUT 3,28 C4COPOUT 3,28 C4COPOUT 3,28 C4COPOUT 3,28 C4COPOUT 3,28 C4COPOUT 3,28 C4COPOUT 3,28 C4COPOUT 3,28 C4COPOUT 3,28 C4COPOUT 3,28 C4COPOUT 3,30 C4FILOAD 3,31 C4PRUNE 3,33 C4STARK 3,33 C4STARK 3,33 C4STARK 3,33 C4STARK	ICEIPUM ICEIPUB, ICEIPUB, ICEKPUB, ICEKPUS, ICEKPUS, ICEKPUA, ICEKPUA, ICEKPUA, ICEKPUA, ICEKPUA, ICEKPUA, ICEKPUA, ICEKPUB, ICEKPUB,	KPUB, OPUB KPUB, OPUB OPUA OPUT OPUT OPUA OPUA OPUA OPUA OPUA OPUA OPUA OPUA	PICK A BVD FOR OUTPUT MAKE KAD ENTRIES CONVERT IRRR TO MBBCCHHR CONVERT MBBCCHHR TO IRRR PARTITION INPUT BLOCKS GET SORTWORK SPACE KEY BVD MERGE KEY INPUT/OUTPUT COUNT RECORDS IN BUFFER COUNT OUTPUT RECORDS IN RSA DUMP OUT INPUT BUFFERS HIGH-TO-LOW SELECT LOAD ROUTINE ALTERNATE LIST PREPARATION SET BLOCK BOUNDS MERGE INPUT LISTS ADD A LIST TO SELECTION TREE SET COMPARE AND BRANCH MODES START/STOP I/O STOP SORTWORK INPUT START SORTWORK OUTPUT

## Licensed Material--Property of IBM Page of LY33-8042-6, As Updated 31 March 1981, By TNL LN20-9345

## EVENT-TABLE DESCRIPTION FOR BLOCKSET, Continued

l										
l	TRACE	SEGMENT	MODULE		DESCRIPTION					
l	3,3A 3,3B	C4TRELL	ICEKPUB ICEKPUB,	ОРИВ	START SORTWORK INPUT TREE LAD LIST RESTORE - HIGH TO LOW					
	3,3C	C4TRELL	ICEKPUB,	OPUB	TREE LAD LIST RESTORE - LOW TO					
	3,3D	C4THIGH	ICEKPUS,	OPUT	PREPARE TREE FOR HIGH TO LOW					
	E0123456789ABCDFF0012345012 340123 33333333333333333333333333333333333	C4PIKBAK C4KYCYLC C4CUTBAK C4FILLUT C4LOADUT C4BACKUT C4KYCYLO C4KYBVDO C4TRKSET C4RUNOUT C4STOUT C4STOUT C4STOUT C4STOUT C4STOUT C4STOUT C4CATA C4WAITIO C4OUTPUT	ICEKPUSTICEOPUJA ICECOPUJA ICECOPUJA ICECOPUJI ICECOPUJI ICECOPUJI ICECOPUJI ICECOPUJI ICECOPUJI ICECOPUJI ICECOPUJI ICECOPUJI ICECOPUJI ICECOPUJI ICECOPUJI ICECOPUJI ICECOPUJI ICECOPUJI ICECOPUJI ICECOPUJI ICECOPUJI ICECOPUJI ICECOPUJI ICECOPUJI ICECOPUJI ICECOPUJI ICECOPUJI ICECOPUJI ICECOPUJI ICECOPUJI ICECOPUJI ICECOPUJI ICECOPUJI ICECOPUJI	KPUT KPUB	TREE LAD LIST RESTORE - LOW TO HIGH PREPARE TREE FOR HIGH TO LOW SELECT PICK BOUNDARY VALUE FOR OUTBACK KEY CYLINDER WRITE BACK TO SORTWORK FILL OUT OUTPUT AREA LOAD SORTOUT BUFFERS WRITE BACKOUT DATA KEY CYLINDER KEY BVD KAD MERGE TEST WRITEBACK LIMIT TRACK SET ALLOCATION SELECT AN OUTPUT RUN LIST STOP SORTWORK INPUT START SORTWORK OUTPUT START SORTWORK INPUT ALLOCATE AND INIT. I/O XFER AREA PERFORM WAIT FUNCTION OUTPUT PASS SORTIN TO RSA BUILD SORTWORK BLOCKS PREPARE TO WRITE TO SORTWORK WRITE SORTWORK BLOCKS SORTIN + E15 TO RSA CALL ICECOBY. E15 TO RSA CALL ICECOBY. E15 TO RSA WRITE SORTED OUTPUT, THEN EMPTY INPUT WRITE SORTED OUTPUT, THEN EMPTY INPUT WRITEBACK, THEN EMPTY INPUT FLUSH OUTPUT WAIT ON BLOCKSET READ EMPTY INPUT BUFFERS ISSUE NEXT BLOCKSET READ EMPTY INPUT BUFFERS ISSUE NEXT BLOCKSET READ MERGE DUMPED RECORDS INTO RSA RUN RESTORE SORTED OUTPUT					
	3,74	C50UTPVT	ICEOPVT		RESTORE SORTED OUTPUT					

Licensed Material--Property of IBM
Page of LY33-8042-6, As Updated 31 March 1981, By TNL LN20-9345

## EVENT-TABLE DESCRIPTION FOR BLOCKSET, Continued

TRACE	SEGMENT		BLOCKSET	DESCRIPTION
3,75	C50UTPVT	ICEOPVT		PARTIAL DUMP MERGE DUMP RECS INT
3,76	C50UTPVT	ICEOPVT		PARTIAL DUMP MERGE DUMP RECS INT INTO RSA RUN PARTIAL DUMP - RESTORE SORTED OUTPUT WRITEBACK TO SORTWORK CALL ICE35VL FOR COBOL RESTORE FLUSH RSA CALL ICE35VL FOR E35 INTERFACE SUPPORT
3,77	CSUTBAKV	ICEOPVI		WRITEBACK TO SORTWORK
3,78	C50UTPVT	ICEOPVT		CALL ICE35VL FOR COBOL RESTORE
3,79	C50UTPVT	ICEOPVT		FLUSH RSA
3,7A	C5RESTOR	ICEOPVA		CALL ICE35VL FOR E35 INTERFACE SUPPORT
			MINIMI - 1 - 1 - 1 - 1 - 1 - 1 - 1 - 1 - 1 -	,
4,4E	C45TARIO	ICEIPUB,	KPUB, OPUB	ISSUE EXCP UPDATE BLOCKSET KAD INDEX (WITH
4,A0	C4KEYMRG	ICEKPUS,	OPUT	UPDATE BLOCKSET KAD INDEX (WITH
4,A1	C4KEYPUT	ICEKUPS,		WRITE HIGH IN-CORE INDEX KADS TO
	0444054	TACTOUT	KOUŤ	DISK TRANSFER AREA INITIALIZATION HANDLE EOF ON SORTIN BEFORE SVC FIND SORTIN MBBCCHHR SET SORTOUT TRANSFER SPACE REGULTREMENTS
4,A1 4,A3	C4XAREA C4EOVIN		KPUI	AVNULE EUE UN CUDIIN BEEUDE GAC
	C4FINDIN			FIND SORTIN MRRCCHHR
4,A6	C4EOF	ICEIPUT		SET SORTOUT TRANSFER SPACE
. •				
4,A7	C4DEBFIX	ICEIPUB,	KPUB, OPUB	FIX DEVICE MODIFIERS BEFORE FIX
4,48		ICEE15B		RETURN FROM E15 WITH RC=8
4,A9	C4COBIN	ICEIPUM		RETURN FROM E15 WITH RC=8
4 , AA		ICEIPUB,	KPUT, OPUB	PUT REAL ADDRESSES IN IOBS, DXLS
4,AB	C40SPACE	ICEKPUT		OPUT SPACE COMPUTATIONS
	C4KSPACE	ICEKPUT		KPUT TRANSFER SPACE COMPUTATIONS
4,AE 4,AF	C4RELSE C4MAINFI	ICEKPUT		RELEASE EXCESS DISK SPACE
4,AF	C4LOAF35	TOERTUV		TRACE OF & DECETHED
4,B1	OTEON 35	ICEIPUT		INITIALIZE FOR INPUT PHASE
4,B2		ICEIPUT		REINITIALIZE AFTER SORTIN EOV
4,B4	C4 INPUT	ICEIPUM		INPUT PASS
4,B5		ICEKPUT		INITIALIZE FOR PHASE 2 AND 3
4,B6		ICEKPUV		VIRTUAL BLOCK CONSTRUCTION
4,B7		ICEEXIK		EXIT FROM KEY PHASE BEFORE
4,88	C4V0BL0K	ICEKPUV		OPUT SPACE COMPUTATIONS KPUT TRANSFER SPACE COMPUTATIONS KPUT TRANSFER SPACE COMPUTATIONS RELEASE EXCESS DISK SPACE PREPARE MIAN STORAGE PRIMARY FIL TRACE RC 8 RECEIVED INITIALIZE FOR INPUT PHASE REINITIALIZE AFTER SORTIN EOV INPUT PASS INITIALIZE FOR PHASE 2 AND 3 VIRTUAL BLOCK CONSTRUCTION EXIT FROM KEY PHASE BEFORE EXITTERM VIRTUAL OUTPUT MERGE BLOCK CONSTRUCTION VIRTUAL INTERUM MERGE BLOCK CONSTRUCTION KAD WRITEBACK MERGE (START OF MODULE)
4,B9	C4VIBLOK	ICEKPUV		VIRTUAL INTERUM MERGE BLOCK
4.BB		ICEKPUS		CONSTRUCTION KAD WRITEBACK MERGE (START OF
.,		- J L K 1 U J		MODULE)
4,BC		TCEKPU2		KAD WRITEBACK SORT HANDLE RECORDS AND INDEXES
4,BD		ICEOPUT		HANDLE RECORDS AND INDEXES

# Page of LY33-8042-6, As Updated 31 March 1981, By TNL LN20-9345

## EVENT-TABLE DESCRIPTION FOR BLOCKSET, Continued

	TRACE	SEGMENT	MODULE	BLOCKSET -	DESCRIPTION
				VLR MODULES	
	4,C0	C5KEYCVL	ICEOPVT		BUILD BLOCKSET KADS FOR WRITEBACK
	4,C1	C5KADCRE	ICEOPVT		BUILD KAD FROM WBACK AREA IN-CORE
	4,C2	C5KADCRE	ICEOPVT		BUILD CYL KAD FOR WRITEBACK AREA
	4,03	C5BAKUTV	ICEOPVA		TERMINATE WBACK LOAD DUE TO RSA
	4,D0		ICEIPVT		ICEIPVT ENTRY
	4,D1		ICEIPVT		ALLOCATE ICEIPVT IOX
	4,D2		ICEIPVT		BUILD ICEIPVT RSA
	4,D3	C5C0DVRT	ICEIPVT		CONVERT ICEIPUT CODE TO BINS IPUM ENTRY OR REENTRY
	4,D4		ICEIPVM		IPVM ENTRY OR REENTRY
	4,D5		ICEIPVM		PRIME SORTIN
	4,D6		ICEIPVM		EOV ON SORTIN - E15
	4,D7		ICEIPVM		EOV ON SORTIN - NO E15
	4,D8		ICEIPVM		REINITIALIZE AFTER EOV - E15
	4,D9		ICEIPVM		REINITIALIZE AFTER EOV - NO SORTIN
	4,DA		ICEIPVM		EOF ON SORTIN - E15
	4, DB		ICEIPVM		EOF ON SORTIN - NO E15
	4,DC		ICEIPVM		RSA IS FLUSHED.
	4,DF		ICEGENV		ICEGENV ENTRY
	4, E0		ICEKPVT		ICEKPVT ENTRY
	4,E1		ICEKPVT		BEGIN ICEKPYT ALLOCATION
	4, E2		ICEKPVT		CREATE SORTED IN-CORE INDEX
	4,E3	C50PINIV			FREEMAIN AND REALLOCATE FOR ICEOPVI
	4,E6		ICEKPVV		BUILD BLOCKSET KADS FROM SKAD
	4,E7		ICEKPVV		CALL KPVS TO SORT SMALL KADS
	4 . E8		ICEKPVV		BUILD BLOCKSET KADS USING KPVS
	•				OUTPUT
	4,EA		ICEKPVV		BUILD BLOCKSET KADS FOR ICEOPVT
	4,F0		ICEOPVT		ICEOPVT TRACE
	4,F1	C50UTPVT			RSA FLUSHED - SORTOUT
	4,F2	C50UTPVT	ICEOPVT		RSA FLUSHED - NO SORTOUT
	4,F9		ICECOBV		RC 8 FROM E15, GOBOL MODE
	4,FA		ICE15V		RC 8 FROM E15, SORTIN CODE
1		C5REXE35			RC 8, SORTOUT MODE
	4,FC	C5REXCOB	ICE35VL		RC 8, COBOL MODE

## Event-Table Description for Peerage and Vale

	<b>p</b> i	EERAGE AND VALE
TRACE	SEGMENT:	DESCRIPTION
1,01	CREMAIN:	TRACE E15 IN
1,01		TRACE E15 IN
1,01		TRACE E15 IN
1,02	CREMAIN:	
1,02	VREMAIN:	
1,02 1,03	CROOUT:	TRACE E15 RETURN TRACE E35 ENTRY
1,03	VPOOTITE.	TPACE E35 ENTPV
1,04	CROOUT:	TRACE E35 RETURN TRACE E35 RETURN
1,04	VROOUT:	TRACE E35 RETURN
1,05	VTMOIIT:	ጥየልሮዊ ዊጓና የለጥየሃ
1,05	LIMOUT:	TRACE E35 ENTRY TRACE E35 ENTRY
1,05	LIVOUT:	TRACE E35 ENTRY
1,06	VIMOUT:	TRACE E35 RETURN TRACE E35 RETURN
1,06	LIMOUT:	TRACE E35 RETURN
1,06 2,3E	PIACOL:	TRACE E35 RETURN WRITE TO WORK FILE
2,3E 2,3F	WEDLAST:	WAIT FOR READ FROM WORK FILE
2,3F	WROUK:	WAIT FOR READ FROM WORK FILE
2,40	WROUK:	WRITE TO WORK FIELD
2,41		WAIT FOR WRITE TO WORK FILE
2,42	CREINIT:	ISSUE A READ
2,42		ISSUE A READ
2,42 2,43		ISSUE A READ CHECK WAIT FOR READ
2,43		CHECK WAIT FOR READ
2,43		CHECK WAIT FOR READ
2,44		ISSUE A WRITE
2,44		
2,44	VROOUT:	ISSUE A WRITE ISSUE A WRITE
2,44	CROOUT:	
2,45	CROEND:	CHECK FOR WRITE
2,45		CHECK FOR WRITE
2,45 2,45	VROOUT:	CHECK FOR WRITE CHECK FOR WRITE
2,45		DATA BLOCK WRITTEN ON AN INTERMEDIATE WORK AREA
2,48	REDSUBG:	EXCHANGE AND WRITE OUTPUT
2,4A		READ FROM WORK FILE
2,4A	VEDAHEAD:	START OF MERGE PASS
2,4B		START OF MERGE PASS
2,4C		START OF A SUBPASS
2,4F		WAIT FOR I/O
2,50		TRACE EMTY/READ
2,52	LIMXWRT:	· · · · · · · · · · · · · · · · · · ·
2,53 2,54	LIMSUBR:	TRACE WAIT TRACE SORTOUT WRITE
2.54	VIMMAIN:	TRACE SORTOUT WRITE
2,54	VIMOUT:	TRACE SORTOUT WRITE
2,54	LIMOUT:	TRACE SORTOUT WRITE
2,54	LIMOUT: LIVOUT:	TRACE SORTOUT WRITE
2,54	VIMLAST:	TRACE SORTOUT WRITE
2,55		TRACE SORTOUT CHECK
2,55	VIMMAIN:	TRACE SORTOUT CHECK

```
----- PEERAGE AND VALE -----
 TRACE SEGMENT: DESCRIPTION
 2,55
 VIMOUT: TRACE SORTOUT CHECK
 2,55
 LIMOUT: TRACE SORTOUT CHECK
 LIVOUT: TRACE SORTOUT CHECK VIMLAST: TRACE SORTOUT CHECK
 2,55
 2,55
 MOVERTBN: PRIMING A NEW STRING
 3,B9
 EOSTRING: END OF STRING PROCESSING
 3.B9
 CRESUBR: BLOCK AN INDEX INTO AN INDEX OUTPUT BUFFER
 3,BA
 3,BB
 PARWRIT1: WRITES A BLOCK OF WRITEVACK INDEXES DURING
 SORT OR FORWARD HASH PASS
 3,BB
 PARWRIT2: WRITES A BLOCK OR WRITEBACK INDEXES DURING
 SORT OR FORWARD HASH PASS
 PARWRIT3: WRITES A BLOCK OR WRITEBACK INDEXES DURING
 3,BB
 FINAL SORT
 3,BC
 PARWRTBO: WRITES A BLOCK OF WRITEBACK INDEXES DURING
 SORT OR FORWARD HASH PASS
 3,BC
 PARWRTB1: WRITES A BLOCK OR WRITEBACK INDEXES DURING
 SORT OR FORWARD HASH PASS
 PARREADO: READ A LOGICAL INPUT INDEX BLOCK
 3,BD
 3,BD
 PARREAD1: READ A WRITEBACK BLOCK
 3,BE
 PARBLOCK: BLOCKS A LOGICAL BLOCK OF INDEXES TO OUTPUT
 3,BF
 PARDBLK: DEBLOCK INDEXES
3,C0 PARMRG: MERGES THE CHAIN OF SORTED INDEXES
3,C1 PARSRT: SORTS A DEBLOCKED LOGICAL INDEX BLOCK
3,C2 REDINIT: TRACE SUBINDEX PRIME
3,C3 REDMAIN: MAIN TRACK ENTRY
3,C4 REDREAD: TRACE TRACK READ
3,C5 REDBACK: GET WRITE BACK
3,C6 REDWRITE: TRACE OUTPUT TRACK
3,C7 REDFLUSH: WRITE LAST INDEX BUFFER AN WAIT ALL
 ACTIVE WORK I/O
 3,CB REDSUBR: FIND RECORD TO START WRITE-BACK
 3,CC REDSUBR: TRACE SEEK PROCESSING
 LIMMAIN: TRACE MAIN ENTRY AND INITIATE READ
 3,CD
 LIMMAIN: TRACE MAIN READ
 3,CE
 LIMBACK: WRITEBACK
 3,CF
 3,D0
 LIMSUBR: TRACE FINDBACK
 3,D1
 LIMSUBR: TRACE INDEX GET
 3,D2 LIMSUBR: TRACE SEEKS
 3,D4 BINALLC: TRACE REPACK IN 3,D5 BINALLC: TRACE REPACK OUT
 3,D6
 VREFLUSH: ENTER USER CODE AT E17
 3,D6
 VREFLUSH: TRACE EXIT IN ENTER USER EOD ROUTINE
 VREMAIN: TRACE EXIT IN ENTER USER EOD ROUTINE
 3,D6
 VREMAINN: TRACE EXIT IN ENTER USER EOD ROUTINE
 3,D6
 3,D6
 VREMININ: TRACE EXIT IN ENTER USER EOD ROUTINE
 3,D6 VREMININ: ENTER USER CODE AT E16
 3,D6 VREMINIT: ENTER USER CODE AT E11
 3,D6 VREMINIT: ENTER USER CODE AT E16
 3,D6
 VROEND: ENTER USER CODE AT E18
 3, D6 VROEND: ENTER USER CODE AT E17
 3,D6
 VROEND: ENTER USER CODE AT E31
 3,D6
 VROEND: ENTER USER CODE AT E37
3,D7
 VREFLUSH: BACK FROM USER CODE AT E17
3,D7
 VREFLUSN: TRACE EXIT OUT
 3,D7
 VREMAIN: TRACE EXIT OUT
 VREMAINN: TRACE EXIT OUT
 3,D7
 3,D7
 VREMININ: TRACE EXIT OUT
 3,D7
 VREMININ: BACK FROM USER CODE AT E16
3,D7 VREMINIT: BACK FROM USER CODE AT E11
3,D7
 VREMINIT: BACK FROM USER CODE AT E16
```

1		P	EERAGE AND VALE
:	TRACE	SEGMENT:	DESCRIPTION
•	3.D7	VPOEND.	BACK FROM USER CODE AT E18
•	3.D7		
•	3,D7	VROEND:	BACK FROM USER CODE AT E17 BACK FROM USER CODE AT E31
	3,D7	AKOEND:	DACK FROM USER CODE AN #37
•	יים אינו 1 אינו	VKOEND:	BACK FROM USER CODE AT E37 TO AVOID EXIT THRU MANEXIT AT I/O TASK
! '	,DE	SORTWOUT:	- Consider the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of the contract of th
!,	1 1579		PUT A MARK IN THE TRACE TABLE
•	, DF		ICECRE ENTRY
	,DF	OD DINIE	ICEVRE ENTRY
	4,E0		TRACE PRIME RSA
	. ,	42/22/22/22/4	TRACE PRIME RSA
	1,E0	VKEMINIT:	TRACE PRIME RSA
	1,E1	CREINIT:	INITIALIZE WORK FILE ROUTINE
	1,E1	VKEMININ:	INITIALIZE WORK FILE ROUTINE INITIALIZE WORK FILE ROUTINE
	1,E1	VREMINIT:	INITIALIZE WORK FILE ROUTINE
	#,E2		RESTORE TREE REGISTER AND GET WORK AREA ADDR
1 :	, E2		RESTORE TREE REGISTER AND GET WORK AREA ADDR
1 :	+ , E3	CREFLUSH:	FLUSH RECORDS REMAINING IN RSA TO WORK FILES ENTER USER EOD ROUTINE, IF DEFINED
! ;	+,E3 +,E3	VREFLUSH:	ENTER USER EOD ROUTINE, IF DEFINED
•	-	VREFLUSN:	ENTER USER EOD ROUTINE, IF DEFINED
	↓, E4		ICEPAR ENTRY
•			READ THE INPUT INDEX FILE
			FINISHES ORDERING OF INDEXES
-	. E7		PROCESSES INDEXES PRODUCED BY PARHSO
•	1,E8	PARSORT1:	ORDER THE INDEXES FOR REDUCTION PHASE
•	1,E9		ICEPAR EXIT
			FLUSH REMAINING PROCESSED INDEXES
	,EB	AEDAKOTe:	ICEVED ENTRY
	, EB	**************	ICERED
	,EC	EXITER (2:	ICEVED EXIT
•	,EC		ICERED EXIT
•	,ED		ICELIM ENTRY
	, ED		ICELIV ENTRY
•	FEE		ICELIM EXIT
	, EE		ICELIV EXIT
•	, EE		ICEVIM EXIT
	EF		ICECRE EXIT
	, EF		ICEVRE EXIT
	, EF		ICEVRN EXIT
	,F0		ICEDEF ENTRY
	, F1		ICEDEC ENTRY
	F2		ICEDEG ENTRY
1 4	,F3		ICEDEV EXIT

#### | FORCING A SPECIALLY FORMATTED DUMP

| This option is only available for Peerage and Vale.

| The default ERETJCL|ERETINV=ABEND|RC16, which was set when Sort was installed, can be overridden in a standard disk sort by the DEBUG | control statement.

| To obtain a specially formatted dump for a sort, the CTRx=value must be specified in the DEBUG statement. This first prints a SNAP dump (corresponding to a normal SYSUDUMP dump), followed by formatted | information as shown in Figure 22.

I The formatted part of the dump is derived from the source code, which is written in the PL/S language. As shown in Figure 23 this is not difficult to interpret, even without a knowledge of PL/S. However, if you would like to know more about the language you can find much information, including how to read source code, in the publication | Guide to PL/S, GC28-6794.

#### | Example:

DEBUG ABEND, CTR 2= 1

SYSTEM DUMP SNAP dump corresponding to a normal SYSUDUMP dump.

### FORMATTED DATA

- 2.1 Save areas The standard save areas used by different levels of the program.
- 2.2 ABEND code A fullword with the format X'xxsssuuu', where xx is the standard ABEND code prefix, sss is the system completion code at program failure (or zeros), and
  - uuu is the user completion code at uncorrectable error (or zeros). This code is equal to zero for definition errors, and equal to the message number for other errors (for example, '046' would represent message ICE046A).
- 2.3 A fullword containing the address of the instruction at which failure occurred.
- 2.4 Register contents when program failure occurred: 16 fullwords giving the register contents in the order 0-15.
- 2.5 Contents of ICECOMMA (sort variables) formatted when program failure occurred, with offsets from Register 13, comments, labels, and definitions.
- 2.6 Trace of important events, in the form

х ууу

where x identifies the part of the program

yyy identifies the segment of code entered.

x can be one of the following codes:

DEF - definition (ICEDEF)

- creation (ICECRE, ICEVRE, ICEVRN)

partitioning (ICEPAR)reduction (ICERED, ICEVED) R

- elimination (ICELIM, ICEVIM, ICELIV, ICEVIP)

- appendage (for PCI, channel end, or end-of-extent)

The first event listed is the most recent\*; the last is the first that occurred (normally DEF ENTRY) .

\*If one of the most recent events listed concerns an exit, the probable cause of program failure is a programming error in the exit routine.

| Figure 22. Contents of a Specially Formatted Dump

## Licensed Material--Property of IBM Page of LY33-8042-6, As Updated 31 March 1981, By TNL LN20-9345

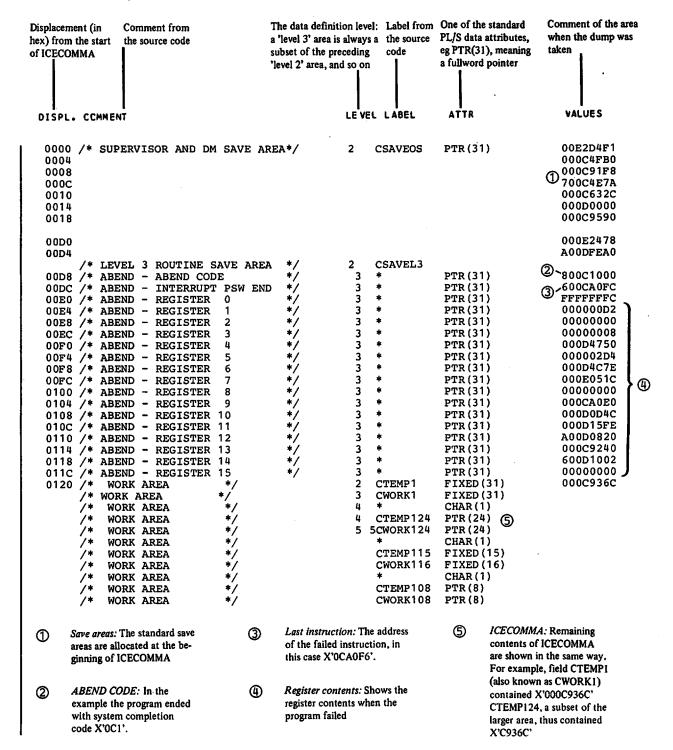


Figure 23. Interpreting a Formatted Dump

Licensed Material--Property of IBM
Page of LY33-8042-6, As Updated 31 March 1981, By TNL LN20-9345

### Finding an Object Module in a Storage Dump

As shown in Figure 24, the name of each module appears at the beginning of the module in the storage dump.

```
LPA/JPA MCDULE
 ICEMON
 C9C3C5D4 E6D54040 09C5D3 40
 ICEMUN REL 5.0 PTF 00 CCPYR1GH
087000
 E340C9C2 D440C3D6 D9D748 140
 *T IBM CCRP. 1973......
087020
 18214100 03004510 CC120 ACA
087040
 CB7060
 45F0CC3C 00087C6C 0000(JCGC
087080
 D507C418 BC90477C CC60' 95E8
 N.M.......
 COFE41AO BU1050AO D44C 95E8

0870A0
CB70C0
 47F0C092 41C0CCC4 CA3C .411C
 0870E0
 D39858F0 D14405EF C20 3D320
 L..CJ...K.L.JH.TGJ....MM.....
 *...CJ...K.L.JF.....
CB7100
 005B58FC 0160C5EF 02()30320
0B7120
 45F0F016 000B712C 00' 000000
 .00.....
 50A00134 18F1502C FC.C00A06
C74F0203 068CF0CB 9'JECFCCC
 027140
0B716C
 ..K.L.C...0...J...M....C....A..C
087183
 C16247FO C18447FO C 1C847FO
 A..CA..CAH.CA8.OB..OB..C42.O....
CB71AC
 1CC4C2JO 100AEOC1 F3FFFD1CC
 0B71C0
 5010D1CC 47F0C196 1.360C10CA
 ..J..OA.U.......A.....OA.N.
 100AC4F4 4770C1C2 C203201C
18F088F0 CG1654F0 C4EC5EFC
OB71E0
 ..D4..ABK.....A..GE....GA.
087200
 .C.C...CL..CC...CC...C...B....
087220
 201041F0 000447F0 C1FC1222
 *...C...CA.....B..O......B..O...
087240
 96402000 45EGC5C8 47F0C226
 *.EH.CB......
CB7260
 20C4BF0F 20044780 C2BA5500
 B...DU..B.P
087280
 41101014 / 41F0F00
 00...^^
0872
 58DD0C01 1
 ••••
 00301
```

Figure 24. The Start of a Module in a Dump

Each module starts with a header consisting of the eight-byte module name followed by the release and PTF number. Some modules have a copyright notice visible after the header.

The module entry point comes after this preliminary material.

With the <u>Blockset</u> technique each load module consists of many object modules. Details are given in Section 3 under 'Phase Structures'. Each object module starts with a header as for the other techniques.

ı

## Origin of Program Messages

This subsection shows which modules issue each message. It also tells you how to locate a given message in the source code. This can be useful if, for example, you encounter an error in Phase 0, and can thus not get a dump.

#### BLOCKSET TECHNIQUE

To find a Blockset message in the code, you should look in the cross reference for labels with the following format:

#### MSG0xxy

where xx is the message number and y is a letter of the alphabet. The letter A means this is the first use of the message in this module, B means it is the second, and so on. For example, MSG046B would be the second occurrence of message ICE046A.

Message	Module	Message	Module	Message	Module
000 010 013 015	EXIN INIB INIB COBV IPVM	046	IPUM IPVM KPUB KPUT OPUA	071	COBU COBV E15B E35B OPVT
017 018 019	15V 35VL INIC INIB EXIN	047 052	KPVT IPUT IPVT EXIO OXOV	073	15V 35VL COBV IPVM 15V
021 023 024 025	INIB INIB INIB EXIO	054 055	EXIO OXOV EXIO OXOV	074 080 081	35VL EXIN EXIO INIC
026	0X0V EXIO 0X0V	056 059 061	INIB INID IPUB	083 .085	DYNA EXIO OXOV
027 039	INIC INIO IPUT IPVT KPUT		IPVB KPUB KPVB OPUB OPVB	088 089 092 093 094	EXIN EXIN EXIN EXIN
042 043	KPVT SUBS MAN IPUM INIB IPVT	063 067	INIB MAN Exin	096 098 099	MESS OXOV IPUM IPVB
	- MAN				

Licensed Material--Property of IBM

#### ALL OTHER TECHNIQUES

To find a Peerage or Vale message in the code, you should look in the cross reference for labels with the following format:

DIAGXXY

where xx is the message number and y is a letter of the alphabet. The letter A means this is the first use of the message in this module, B means it is the second, and so on. For example, DIAG46B would be the second occurrence of message ICE046A.

Exceptions to the above rule occur in modules ICEDEC and ICEDED, which also use tables for issuing messages. There you should look for labels:

SERRXXY or SNERRXX

where xx is the message number, and y is the letter L or P. other Message Cross-Reference Tables

	Technique->	 	Disk							Tap	е			Merge
	Message	PEER	VALE	CR	CX	BA	LN	OS	CL	PO	LY	BA	LN	-only
į	V			fix	var	fix		fix		fix		fix	var	ii
1	000	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC
	001	DEC		DEC	DEC	DEC	DEC	DEC		DEC	DEC	DEC	DEC	DEC ]
	002		. – –	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC
	003			DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC
	004	•	•	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC
	005	•	•	DEF	DEF	DEF	DEF	DEF	DEF	DEF	DEF	DEF	DEF	DEF
ii		•	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC
· i	006			DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC
	007	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC 1
	008	•		DEC 4	DEC <sup>1</sup>	DEC 1	DEC 5	DEC 1	DEC 7	DEC 1		DEC <sup>a</sup>	DEC <sup>4</sup>	DEC
	010	DEC8	DECa	DEC	DECe	DECe	DEC 8	DECs	DEC 8	DEC8	DEC 8	DECe	DEC8	DECe
	012			DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC
	013		DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC
į	014	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC
1 i	015 no E	i - :	VRE2	j -	RDS	i -	RDG	i -	RDS	1 -	RDG	<b>!</b> -	RDG	-
ii	Exits	i -	VRE2	j -	RDT	i -	RDE	<b>-</b>	RDT	] -	RDE	1 -	RDE	-
Ť	016	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC
Ιİ	017	DED	DED	DED	DED	DED	DED	DED		DED		DED	DED	DED
ii		DEG	DEG	DEG	DEG	DEG	DEG	DEG	DEG	DEG	DEG	DEG	DEG	DEG
Ì	018	DEC	DEC	DEC	DEC	DEC	DEC	DEC		DEC		DEC	DEC	DEC
į	020	DEC	DEC	DEC	DEC	DEC	DEC	DEC		DEC		DEC	DEC	DEC
-1-1	021	DECe	DEC	DECe	DECe	DECe	DEC 8	DECa		DEC		DECe	DECe	DECe
11	022	DECe	DECB	DEC	DECs	DECe	DEC <sub>8</sub>	DECe		DEC		DECs	DECe	DECe
İİ				DEC	DEC8	DECe	DEC 8	DEC®		DEC		DECe	DECe	DECal
1 1	024	DECa	DEC8	DECe	DECs	DEC8	DEC 8	DECs	DEC a	DEC	DECa	DECe	DEC8	DECal
1 1		CRO	VRO	1 -	-	-	-	-	-	-	-	1 -	-	- 1
1 1		•	VIM	-	- [	<b>-</b>	-	<b>!</b>		<u> </u>	_		-	-
	027	DEC8	DECB	DECe	DECe	DECa	DEC 8	DECe		DEC		DECe	DEC	DECe
	028	1 -	-	RCH •	RCH 4	RCH 4	RCH 4	RCH <sup>1</sup>	RCH 1	RCH •		RCH <sup>4</sup>	RCH4	RCH
	029	DEC		DEC"	DEC4	DEC"	DEC4	DEC4		DEC4		DEC4	DEC*	DEC
	030	DEC	DEC	RCH	RCH	RCH	RCH	RCH	RCH	RCH	RCH	RCH	RCH	RCH
	031	<b>! -</b>		RCH	RCH	RCH		RCH	RCH	RCH	RCH	RCH	RCH	RCH
	032	DEC	DECe	RCH	RCH	RCH	RCH	RCH	RCH	RCH		RCH	RCH	RCH
	033	DECe	DECB	RCH	RCH	RCH		RCH		RCH	RCH	RCH	RCH	RCH
	034	DEC		DEC	DEC	DEC	DEC	DEC	DEC	DEC		DEC	DEC	DEC
	035	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC		DEC	DEC	DEC
	036	! -	•	BGB	BGB	RCK	RCK	RCS	RC S	RCS		RCS	RCS	- !
	037	! -	] -	BGB	BGB	RCK		RCS	RCS	IRCS		IRCS	RCS	! - !
	038	<u>  </u>	<u>  -                                   </u>	RCJ	RCJ	RCJ	RCJ	RCS	RCS	RCS	RCS	RCS	RCS	DOT I
			. – – .	BGB	BGB	RCK		RCS	RCS	RCS	RCS	RCS	RCS	RCL
ij			VRE2	! -	-	-	-	! <del>-</del>	_	-	_	-	_	-
Ţ	_	PAR	-	] -	-	-	-	-	_	! -	_	! <del>-</del>		_
!!		•	VIP	-	-	! -	-	! -	_	! -	_	-	_	!
ı	all	MON	MON	-	DOTE	- DOTS	DOTS	I PCT	RCI	RCI	RCI	RCI	RCI	
	040	! -	-	RCIS	RCI5	RCIS	RCI5	RCI	KC T	L	VCT.	1 VCT	VCT	_
	041	I DECS	IDECS	RCJ	RCJ	RCJ DEG <sup>7</sup>	RCJ DEG <sup>7</sup>	I – I DEG7	DEG <sup>7</sup>	DEG <sup>7</sup>	DEG7	DEG7	DEG <sup>7</sup>	DEG7
1 1	042	DEG <sup>5</sup>	DEG <sup>5</sup>	DEG7	DEG?	r	DEG.	r	DEG.					1 25G

<sup>\*</sup>Or RCD (if the error is in a MODS statement parameter)

<sup>2</sup>Or VRN

<sup>|</sup> Or LIV "Or RCH

<sup>5</sup>RCJ if the error is that insufficient tracks were allocated (rather than an insufficient number of devices)

For RCI, DEF, or DEC

Or DED

	Technique->	Disk						Tape						Merge
		PEER	VALE	CRO	x	BA		0.50		PO		BAI		-only
	V	ļ	1	fix	var	fix	var	fix	var	fix	var	fix	var	 
!	043	DEC	DECS	DED9	DED <sub>9</sub>	DED9	DED 9	DED®	DED 9	DED®	DED 9	DED®	DED9	DED9
1	044 ph0	DEC44	DEC44	- 9GN	9GN	AGI	- AGI	AGN	AGN	AGA	AGA	AGA	AGA	_
	ph1 ph2		-	I 9GN	9GN	AGJ	AGJ	AGN	AGN	AGG	AGG	AGG	AGG	-
1	ph2	_	_	AGK	AGK	AGK		APK		APK		APK	APK	APF
	045	i	-	l -	-	RPC	RPC			RPC	RPC	RPC	RPC	-
	046 all	MON	MON	i _	_	1 -	_	l -	_	-	_	_	_	i - i
1	ph1	CRE	VRE3	80N	80N	ROK	ROK	RON	RON	ROJ	ROJ	ROI	ROI	i – i
	ph2	RED	-	80N	80N	-	_	RON	RON	ROS		ROR	ROR	i - i
i	047 ph1	CRE	VRE2	80N	8ON	RPC	RPC	RON	RON	RPC	RPC	RPC	RPC	i - i
1	• · · · <b>p</b> ·	CRO	VRO	-	-	-	_	_	_	i -	_	-	-	i – i
i	ph2	i -	-	8PM	8PM	RPF	RPF	RPM	RPM	RPF	RPF	RPF	RPF	i – i
•	ph3	i -	<b>i</b> –	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG
	048	i -	j –	RDR	RDS	RDD		RDR	RDS	RDD	RDE	RDD	RDE	i – i
	049	i -	<b>i</b> -	-	_	RPC	RPC	i -		RPC		RPC	RPC	i - i
	050	i –	j –	8PM	8PM	RPF	RPF	RPM	RPM	RPF	RPF	RPF	RPF	i - i
	051	i -	VED	i -	_	<b>i</b> –	-	i -	_	i -	-	ROR	ROR	i - i
1	052	i -	<b>i</b> –	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG
i	053	MON	MON	ROQ40	ROQ40	ROQ 40	ROQ .O	ROQ	ROQ 40	ROQ10	ROQ4 O	ROQ10	ROQ4 O	i - i
i	054	LIMa	VIM	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG
i i		CRO	VRO	i <b>-</b>	-	i -	-	i -	-	j -	-	i -	-	i - i
i i	055	LIM3	VIM	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG
İ		CRO	VRO	-	-	-	-	-	<b>-</b>	<b>i</b> -	-	-	- ;	-
i	056	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC	DEC
1		DED	DED	RCI	RCI	RCI	RCI	RCI		RCI	RCI	RCI	RCI	RCI
1	057	i -	-	-	-	į -	-	RCI		RCI	RCI	RCI	RCI	-
	058	1 -	-		-	-	-	RCI		RCI	RCI	RCI	RCI	-
i	059	DEC	DEC	DEC.	DEC <sub>0</sub>	DECe	DEC <sub>9</sub>	DECa	DEC 9	DEC	DEC.	DECa	DEC	DEC.
- 1	060	-	] -	RC4	RC4	RC4	RC4			<u>  -                                   </u>		_ <b>_</b> _	-	-
	061 ph0	-	-	RCB	RCB	RCB	RCB	RCB		RCB	RCB	RCB	RCB	RCB
ļ.	ph1 read	MON a		RCV	RCV	RCV		RCV	RCV	RCV	RCV	RCV	RCV	RCV
ļ ļ	write	:	MON 4	8PA	8PA	RPB	RPB	RPA	RPA	RPA	RPA	RPA	RPA	- 1
Ţ	ph2 rd b*5	-	-	<del>-</del>	-	-	-	RGB 4 3	KGB 4 3	RGB 43		RGB <sup>4 3</sup>	RGB4 3	-
! }	rd fos	MON	MON	8GB	8GB	RGC 43	RGC 4 3	•	-	RGL	RGL43	•	RGL 3	_
! !	write	MON	MON	8PA	8PA	RPE	RPE	RPA	RPA	RPD	RPD	RPD	RPD	- !
!	ph3 rd b45	-	-	-	00013	-	DCD43	RGD 43		RGD 43	RGL43		RGD <sup>4.3</sup>	
!	rd f <sup>45</sup>	•	MON4 4	8GC 1 3		RGE 43	RGE 43	•	DCII	RGM43			RGM <sup>4 3</sup>	•
!	write  062	MON44	WON4	RCV	RCV	RCV	RCV RCO	RCV RCO	RCV RCO	RCV	RCV RCO	RCV RCO	RCV RCO	RCV
1	063 ph0	DEF 4	DEC 2	RCO RCM <sup>4</sup>	RCO RCM 4	RCO RCM 4 4	RCM 44			RCO		RCM <sup>®</sup>	RCM <sup>4</sup>	RCM <sup>4</sup>
		CRE	•	APC	APC	APC	APC	APC	APC	APC	APC	APC	APC	-
	ph1	CRO	VRO	-	-	-	_	-	-	-	_	-		- 1
'	ph2	-	-	i -	-	APJ	APJ	-		APJ	APJ	APJ	APJ	-
	ph3	-	i -	AGH	AGH	AGH	AGH	AGH		AGH	AGH	AGH	AGH	AGF
	064	i –	- '	RCV	RCV	RCV		RCV		RCV	RCV	RCV	RCV	RCV
	065	DEC	DEC	RCH	RCH	RCH		RCH	RCH	RCH		RCH	RCH	RCH
1	2Or VRN 3Or LIV 6Or DEF or 1 6Or DED 9Or RCI 10ROP if mu: 11Or VRE, VI	ltiple RN, VR	O, XCP	re rou	tine u	sed		Lacas		<b></b>				
1	130r the co			assiqn	ment me	odule								i
•	440r DEC or	DED	·											i
	45rd b mean		backwa	ard (ta	ape),	rd f me	eans r	ead for	rward	(tape)	, read	(disk)	)	i

Licensed Material -- Property of IBM Page of LY33-8042-6, As Updated 31 March 1981, By TNL LN20-9345

Tech	rechnique->   Disk							<u> </u>		Iap	e 			Merg
Mess	age	PEER	VALE		CX		LN		CL		LY	BA		-onl
v		!	<u> </u>	fix	var	fix	var	fix	var	fix	var	fix	var	] L
066	ph1	-	-	80N	80N	ROK	ROK	RON	RON	ROJ	ROJ	ROI	ROI	-
	ph2	i -	i -	80N	80N	j -	-	RON	RON	ROS	ROS	ROR	ROR	1 -
067	-	DEF	DEF	DEF	DEF	DEF	DEF	DEF	DEF	DEF	DEF	DEF	DEF	DEF
068		<b>-</b>	-	i -	-	j –	-	j <b>-</b>	-	-	-	-	-	ROQ9
069	'single'	<b>-</b>	i -	ROB	ROD	ROB	ROD	ROB	ROD	ROF	ROH	ROB	ROD	BALN
" mt	ltiple'	-	i –	ROA	ROC	ROA	ROC	ROA	ROC	ROE	ROG	ROA	ROC	(as
070	-	-	j -	j -	-	RCK	RCK	j -	-	-	-	-	<b>-</b>	j -
071	ph1	CRO	VRE2	RDR	RDS	RDD	RDE	RDR	RDS	RDD	RDE	RDD	RDE	j -
		i -	VRO	j -	_	i -	-	i -	-	<b>-</b>	-	-	-	<b>i</b> -
	ph2	i - i	VEE	RBW	RBX	RBJ	RBK	RBW	RBX	RBJ	RBK	RBJ	RBK	j -
	ph3	LIM48	VIM	RBM	RBO 16	RBM	RBO 16	RBM	RBO 46	RBM	BBO4 e	RBM	RBQ16	RBM <sup>4</sup>
072		DED	DED	i -	_	-	-	i –	-	i <b>-</b>	-	i -	4	j -
		-	DEC	i -	-	j -	-	-	_	-		i -	-	i -
073	no E	-	VRE2	i -	$\mathtt{RD}\mathbf{T}$	<b>.</b> –	RDG	i _	RDT	-	RDG	i -	RDG	i -
• • •	Exits		VRE2	i -	RDS	<b>;</b>	RDE	i -	RDS	_	RDE	-	RDE	i -
074	DA T CO	•	DEC8	DEC	DEC <sub>9</sub>	DEC	DEC 9	DEC9	DEC 9	DEC	DEC 9	DEC9	DEC <sub>9</sub>	DEC <sub>2</sub>
0751	8	•	ERR	-	-	, <u>, , , , , , , , , , , , , , , , , , </u>	-	-		-	_	-	_	-
076		ERR	ERR	RGV	RGV	RGV	RGV	RGV		RGV	RGV	RGV	RGV	RGV
077		ERR		I RPV		RPV		RPV		RPV	RPV	RPV	RPV	RPV
	ph049	ERR	ERR	RCI	RCI	RCI	RCI	RCI		RCI	RCI	RCI	RCI	RCI
070	ph1		- LEKK	•	APC	APC	APC	APC	APC	APC	APC	APC	APC	-
		-	<del>-</del>	APC	APC	I APC	APC	APC	APC -	AFC	AFC .	I -	nrc -	l AGF9
Ьщэ	in	-   ERR	ERR	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG
070	out			I RPG	KPG -	KPG	-	KFG	-	KFG	REG	L EG	KrG	A-
079		ERR	ERR	1		RCT	RCT	5	RCT	RCT	RCT	RCT	RCT	-
ph1	:20	-	! -	RCT	RCT	KCT	RCT	RCT	RCT	I TO I	RC1	LCI	KC1	RGF
	in <sup>20</sup>	[ - ]	! -	-	-		-				_	•	- nnc	
ph3	out20	-	-	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG	RPG
080		CRO	VRO	! -	-	! -	-	-	-		-	-	_	-
081		DEV	DEV	<u> </u>	-	! -	-	-	-	-	-	-	-	] -
082		•	VED	ļ -	-	! -	-	•	-	-	-	-	_	-
083		•	DEG	-	-	! -	-	-	-	-	-	. <del>-</del>	-	! -
084			DEV	! -	-	] -	-	-	-	-	-	-	-	! -
085		DEC	DEC	! -	-	-	-	_	-	_	-	_	•	] -
ph1		•	VRO	-	-	} -	-	-	-	-	-	-	-	1 -
ph3			VIM	1 -	-	-	-	-	-	-	-	-	-	-
<u>a</u> 11	•	MOM	MON	1 -		1 -	-	-	-	L -	- :	_	-	l -
087			MON	! -	- 1	-	-	-	-	<b>-</b>	-	-	-	-
880		DEV	DEV	j -	-	1 -	-	-	-	-	-	-	-	-
089			DEV	j -	-	-	-	-	-	-	-	-	-	-
090	j	DEC	DEC	j –	<b>-</b>	-	-	-	-	-	-	-	-	i -
091		MON	MON	i -	- 1	-	-	-	-	-	-	-	-	i –
092		DEV	DEV	i -	-	<b>-</b>	-	-	-	<b>-</b>	-	<b>-</b>	-	i –
093		DEV	DEV	i -	-	i -	- j	<b>-</b>	-	_	-	-	-	i –
099		XCP	XCP	-	_	i _	_	_	_ i	_	_	i 🕳	_ 1	i _

<sup>&</sup>lt;sup>2</sup>Or VRN <sup>3</sup>Or LIV

eor DED

Or RCI

<sup>168</sup>BO if records are spanned

<sup>17</sup>RBO if variable non-spanned records, 8BO if spanned
18Can be issued by any of a large number of modules, including all those which generate control blocks, open, close, and block output. Look in the dump at the address given in the message.
19With a conventional technique can be RCI, DED or RCZ

<sup>20</sup>RCT if a sort terminates unsuccessfully

### Cross-Reference Tables

#### CPI-PPI CROSS-REFERENCE TABLE

The following table contains the names of CPI and PPI fields together with their displacements; the last three letters of the names of modules that reference them; and the manner in which the modules are related to them. A dash is used to separate the module name from the related code. The codes are as follows:

- U The information in the field is  $\underline{u}$  sed by the module.
- M The information in the field is created or  $\underline{m}$  odified by the module. It may also be  $\underline{u}$  sed.
- O Non-specific relationship exists between the module and the field. Either or both of the <u>o</u> ther two codes may, or may not, be applicable.

Field	Displacement	Addres	sed by	Modules	******			
CP IADDCF	(EB)	RCI-U	RCM-M	RCN-U	RC1-U	RC2-U		
<b>CPIAQTB</b>	(133)	RCM-M	RC 1-U					
CPIATP1E	(C7)	RCI-U	RCM-M	RC 1-U				
CPIATP3E	(CD)	RCI-U	RCL-U	RCM-M	RC1-U			
<b>CPIAUENT</b>	(112)	RCH-M	RC 1-U					
CPIBINSZ	(C2)	BGA-U	BGB-U	RCN-M	RCR-U	RCS-U	RC1-U	
CP IBUF1	(E5)	BGA-M	BGB-M	RC1-U				
CPIBUF23	(AE)	BGA-M	BGB-M	RCL-M	RC1-U			
CPIDCBIN	(FA)	BGB-U	RCI-M	RCL-U	RCS+U	RC 1-U	8CI-M	
CP IDCBOU	(FC)	BGB-U	RCI-M	RCL-U	RCS-U	RC1-U	8CI-M	
CPIDDSRT	(F6)	RCI-U	RCM-M	RC1-U				
CP IDEV	(102)	BGB-U	RCI-M	RC 1-U				
CP IDEVCD	(132)	RCI-M	RC 1-U					
CPIFFF	(A4)	RC 1-U	RC 2-M					
CPIFILSZ	(D3)	RCM-M	RCS-U	RC 1-U	8CI-M			
CPIIPBLK	(A2)	BGB-U	RCI-M	RCS-U	RC1-U			
CP IJO BNM	(11F)	RCM-M	RCZ-U	RC <b>1-</b> U				
CPILAB03	(BC)	BGA-M	BGB-U	RCI-M	RCL-U	RCS-U		
CPILAB07	(BE)	BGA-M	BGB-M	RCL-M	RCS-M	RC 1-U	RC2-M	
CPILAB09	(9A)	BGA-M	BGB-M	RCI-M	RCL-M	RCS-U	RC 1 – U	
CP IL INK	(E8)	RCH-M	RCM-U	RCO-M	RCZ-U	RC1-U		
<b>CP IMODEX</b>	(CA)	BGA-U	BGB-U	RCH-U	RCI-M	RCL-U	RCM-M	RCS-U RC1-U
CP IMRGAL	(E7)	BGA-M	BGB-M	RCI-M	RCL-U	RCM-M	RC1-U	
CP IMRGM X	(E6)	BGA-M	BGB-M	RCL-M	RC1-U			
CP IMSGDD.	(116)	RCB-U	RCM-M	RCO-M				
CP IMX COL	(130)	RC 1-U	RC2-M					
CP INUMCF	(48)	RCI-U	RCM-M	RCN-U	RC1-U	RC2-U		
CP INWKU	(9E)	BGB-U	RCI-M	RCL-U	RCN-U	RCS-U	RC1-U	
CP IOPBLK		RCI-M	RC 1-U					
CPIPBUFF	(A6)	RC 1-U	RC2-M			_		
CPIPCF01	(4A)	RCI-O	RCM-M	RCN-O	RC1-U	RC2-0		
CP IPGCSZ	(C0)	BGB-U	RCL-U	RCS-U	RC1-U	RC2-M		
CPIP1ASZ	(A0)	RC 1-U						
CP IP 1GC	(DC)	BGA-M	BGB-M					
CP IP 1RSZ	(DC)	BGB-U	RCI-M	RCS-U	RC1-U			
CP IP2GC	(DF)	BGA-M	BGB-M					
CPIP2RSZ	(DF)	BGB-U	RCS-U	RC 1-U				
CP IP3ASZ	(AA)	RCB-U	RCM-M	RCZ-U	RC1-U			
CPIP3GC	(E2)	BGA-M	BGB-M					

Field	Displacement	Addre	ssed by	Modules	5 				
CPIP3RSZ	(E2)	BGB-U	RCI-M	RCL-M	RCS-U	RC 1-U			
CP IRCDL 1	(B2)	BGA-U 8CI-U	BGB-U	RCI-M	RCL-U	RCM-M	RCR-U	RCS-U	RC1-U
CP IRCDL2	(B4)	BGA-U	BGB-U	RCI-M	RCL-U	RCM-M	RCN-U	RCR-U	RCS-U
CP IRCDL3	(B6)	BGA-U		RCI-M	RCL-U	RCM-M	RCS-U	8CI-U	
CP IRCDL4	(B8)	RCI-M	RCM-M	RCN-U					
CP IRCDL5	(BA)	BGB-U	RCI-M	RCM-M	RCN-U	RCR-U	RCS-U		
CP IRECFM	(11E)	RCI-M	RC 1-U						
CP ISESDS		8CI-M							
CPISIRG	(D9)	BGA-M	RC 1-U						
CPISKPRD	(C4)	RCM-M	RC1-U						
CPISKSDS CPISMAX		8CI-M RCI-U	RCM-M						•
CP ISORCE	(EE)	RCH-M	RC1-U						
CPISRTBL	(BD)	BGA-M	BGB-M	RCI-M	DCS-II	RC 1-U			
CPISRTG		BGA-M	BGB-M	RC1-U	NCD 0	MC 1-0			
CP ISTPNM	(127)	RCM-M	RCZ-U						
CPISUENT	(115)	RCH-M	RC 1-U	RC2-U					
CPISVSAM	( ,	RCI-U	8CI-U						
CPISVSIN		BGB-U	RCI-U	RCL-U	RCS-U	8CI-M			
CPISVSOP		BGB-U	RCI-U	RCL-U	RCR-U	RCS-U	8CI-M		
CP ISW1	(92)	BGA-M	BGB-M	RCD-M	RCH-M	RCI-M	RCL-M	RCM-M	RCN-N
		RCP-U	rcs-u	RCZ-U	RC1-U	RC2-U	8CI-M		
CPISW14		BGB-U	RCI-M	RCN-U					
CP ISW17		RCN-M							
CP ISW2	(12F)	BGB-U	RCI -U	RCL-U	RCM-M	RCO-U	RCR-U	RCS-U	RCZ-U
	•	RC 1-U	8CI-M		_				
CP ITAVLC	(D0)	BGA-M	BGB-M	RCL-U	RCM-M	RCS-U	RC 1-U		
CP IXCAP	(F8)	BGA-M	RC 1-U						
PPIADDCF	(4 18)	AOM-U	RCZ-U	RC1-M	3 DN7 11				
PP IADSSC	(324)	ABF-U	ABL-U	ABM-U	ABN-U	ABO-U	ABP-U		ABR-U
		ABS-U AGJ-U	AGA-U AGK-U	AGC-U AGL-U	AGD-U AGM-U	AGE-U AGN-U	AGF-U AOA-U	AGG-U AOB-U	AGI-U
		AGD-U	AGR-U AOE-U	AGL-U	AGM-U AOG-U	AGN-U AOH-U	AOA-U AOM-U	APA-U	APB-U
		APD-U	ADE-U	APF-U	APG-U	APH-U	API-U	APK-U	APL-U
		RBJ-U	RBK-U	RBM-U	RBO-U	RBW-U	RBX-U		RCV-M
		RC6-U	RC7-U	RC8-U	RDD-U	RDE-U	RDG-U		RDS-U
		RDT-U	RDU-U	RGB-U	RGL-U	RGM+U	RGV-U	ROI-U	ROJ-U
		ROK-U	RON-U	ROP-U	ROQ-U	ROR-U	ROS-U	ROT-U	RPA-U
		RPB-U	RPC-U			RPF-U	RPG-U	RPM-U	
		8BO-U	80N-U	9BN -U	9BO-U	9GB-U	9GC−U	9GN-U	
PP IALG	(328)	AOI-U	U-Loa	AOK+U	AON-U	AOR-U	AOS-U	U-TOA	RCT-O
		RC6-O	RC7-M	RC8-M	RC9-0	RDR-U	RDS-U	RGB-U	RGC-Ü
		RGL-U	ROW-U	ROY-U	RPA-U	RPB-U	RPD-U	RPE-U	RPG-U
		8GB-U	8PA-U	90N-U				<b></b>	
PPIAMA	(3A9)	ABF-U	ABS-U	AGA-U	AGI-U	AGN-U	AOA-U	AOB-U	AOC-U
		AOD-U	AOE-U	AOF-U	AOG-U	U-HOA	APA-U	APB-U	APC-U
PPIAMB	(3A8)	APG-U	APL-U AGB-U	RC9-O AGC-U	9GN-U	9PA-U	AGL-U	ACM_II	3 DD_11
r c Tund	(avo)	ABR≁U APE-U	AGB-U APH-U	AGC-U APJ-U	AGG-U APL-U	AGJ-U 9GB-U	WGT_0	AGM-U	APD-U
PPIAMC	(3A8)	ABL-U	ABM-U	ABN-U	ABO-U		ABQ-U	AGD-U	AGE-U
	()	AGF-U	AGH-U	AGK-U	APF-U	API-U	APK-U	9BN-U	9BO-U
		9GC-U			7			<i>,</i> , <i>,</i>	<i></i>
PP IAPGC	(432)	RCB-U	RC 1-M	RC6-U	RC7-U	RC8-U			
PPIATP1E	(54)	ADD-U	ADE-U	ADR-U	ADS-U	ADU-U	AOQ-U	APC-U	APF-U
	: •	RC 1-M	RC6-U	="			- <b>-</b> -	-	
PPIATP3E	(2C4)	ABM-U	ABO-U	APF-U	RC1-M	RC8-U	RPG-U	9B0-U	
PPIAXERT	(29C)	AOA-U	AOB-U	AOC-U	AOD-U	AOE-U	AOF-U	AOG-U	AOH-U
		AOL-U 80N-U	AOM-U	AOP-U	U-QOA	RC 1-M	ROW-U	ROY-U	RPC-U

Field	Displacement								
PP IBDSVA	(2E8)	ABC-U	ABF-U	ABH-U	ABJ-U	ABQ-M	ABR-M	ABS-M	ABT-U
		ABU-U	ABV-U	ABW-U	ABX-U	ABZ-U	ADE-U	ADG-U	ADH-M
		ADI-M	ADL-M	ADS-M	ADT-M	ADX-M	AGA-U	AGB-U	AGC-U
		AGD-U	AGE-M	AGG-U	AGH-U	AGL-U	AGM-U	AGN-U	AOI-M
		AOJ-M	M-NOA	AOQ -U	APA-U	APC-M	API-M	APJ-U	APK-U
		RBA-M	RBC-U	RBE-M	REG-U	RBH-U	RBI-U	RBJ-U	RBK-U
		RBM-U	RBP-U	RBT-U	RBU-U	RBV-U	RBW-U	RBX-U	RBZ-U
		RCT-U	RDC-U	RDD-U	RDE-M	RDG-M	RDI-U	RDL-U	RDQ-U
		RDR-U	RDS-M	RDT-M	RGB-U	RGL-U	ROK-M	RON-U	ROS-U
		RPC-U 90N-M	RPF-U	RPM-U	80N-U	9BN-U	9BO-U	9GB−U	9GC−U
PPIBINSZ	(260)	ABF-U	ADE-U	ADG-U	ADP-U	ADQ-U	ADS-U	ADT-U	APC-U
	(200)	APG-U	APL-U	RCK-U	RC1-M	RDB-U	RDC-U	RDD-U	RDP-U
		RDQ-U	RDR-U	RDS-U	RDT-U			1.55	
PPIBLK	(340)	ABA-U	ABB-U	ABC-U	ABE-U	ABL-U	ABM-U	ABN-U	ABO-U
	(0.00)	ABP-U	ABY-U	ABZ-U	ROA-U	ROB-U	ROC-U	ROD-U	ROE-U
		ROF-U	ROG-U	ROH-U	ROP-U	ROQ-U	RPA-U	RPB-U	RPG-U
		8PA-U	9BN-U	9BO-U					
PPIBLK2	(370)	ABG-U	ABH-U	ABI-U	ABJ-U	ABK-U	ABT-U	ABU-U	ABV-U
	• - •	ABW-U	ABX-U	RGB-U	RGC-U	RGL-U	ROP-U	ROQ-U	RPA-U
		RPD-U	RPE-U	8GB-U	8PA-U			~	
PPIBPTRK	(2AA)	AGC-U	AGE-U	RCJ-M	ROK-U	RPB-U	RPE-U	8CK-M	8GB-U
	•	9PA-U							
PPIBUF1	(5C)	APB-U	APG-M	APL-U	RCJ-U	RCK-M	RC1-M	8CK-M	
PPIBUF23	(274)	APH-U	API-U	APL-U	RCK-M	RC1-M	8CK-U		
<b>PPICHKAD</b>	(424)	AGH-U	APC-U	APF-U	APJ-U	RC9-M	RON-U	ROS-U	ROT-U
PPICONV	(380)	ABF-U	ABL-U	ABM-U	ABN-U	ABO-U	ABP-U	ABQ-U	ABR-U
		ABS-U	AGA-U	AGB-U	AGC-U	AGD-U	AGE-U	AGG-U	AGI-U
		AGJ-U	AGK-U	AGL-U	AGM-U	AGN-U	U-AOA	AOB-U	AOC-U
		AOD-U	AOE-U	AOF-U	AOG-U	AOH-U	APA-U	APB-U	APD-U
		APE-U	APF-U	APG-U	APH-U	API-U	APK-U	APL-U	RC6-M
		RC7-U	RC8-M	RC9-U	ROI-U	ROJ-U	ROK-U	RON-U	80N-U
		9BN-U	9BO-U	9GB-U	9GC−U	9GN−U	9PA-U		
PPICOUNT	(240)	ABG-M	ABH-M	ABJ-M	RBG-M	RBH-M	RBI-M	RBJ-M	RBK-M
•		RBL-M	RBM-M	RBN-M	RBO-M	RBP-M	RDB-M	RDC-M	RDD-M
		RDE-M	RDG-M	RDP-M	RDQ-M	RDR-M	RDS-M	RDT-M	ROA-U
		ROB-U	ROC-U	ROD-U	ROE-U	ROF-U	ROG-U	ROH-U	ROI-U
		ROJ-U	ROK-U 8BN-M	RON-U	ROR-U	RPC-M	RPE-U	RPF-M	RPG-U
PP ICPTRK	(# 20)	RPM-M		8BO-M	80N-U	8PM-M			
PPIDCBIN	(428)	RCJ-M AGA-U	8CK-M AGI-U	8PA-U AGN-U	9GN-U APF-U	RCK-U	RC1-M	9GN−U	
PPIDCBOU	(42A)	AGK-U	APF-U	APK-U	RCK-U	RC1-M	KC I-M	3GN-0	
PPIDDSRT	(41C)	AGA-U	AGG-U	AGH-U	AGI-U	AGJ-U	AGK-U	AGN-U	APC-U
I I I I DOOKI	(410)	APF-U	APJ-U	APK-U	RC1-M	RC4-U	RPG-U	9GN-U	MFC-0
PPIDDOL1	(298)	AOA-U	AOB-U	AOC-U	AOD-U	AOE-U	AOF-U	AOG-U	AOH∸U
	(230)	AOP-U	AOQ-M	RC1-M	AOD O	ACL C	AQI U	100 0	11011 0
PPIDEB	(330)	ABC-U	ADB-U	ADC-U	ADD-U	ADE-U	ADG-U	ADH-U	ADI-U
	(550)	ADJ-U	ADP-U	ADQ-U	ADR-U	ADS-U	ADI-U	ADU-U	ADX-U
		AGA-U	AGI-U	AGN-U	RBA-U	RBB-U	RBC-U	RBE-U	RBL-U
		RBM-U	RBN-U	RBO-U	RBP-U	RBY-U	RBZ-U	RGD-U	RGE-U
		RGM-U	ROA-U	ROB-U	ROC-U	ROD-U	ROE-U	ROF-U	ROG-U
		ROH-U	ROI-U	ROJ-U	ROK-U	RON-U	ROW-U	ROX-U	ROY-U
		ROZ-U	8BN-U	8BO+U	8GC-U	80N-U	9DJ-U	9GN-U	
PPIDEB2	(360)	ADL-U	RGB-U	RGC-U	RGL-U	8GB-U			
PPIDELCT	(244)	RBJ-M	RBK-M	RBW-M	REX-M	RDE-M	RDR-M	RDS-M	ROA-U
	,	ROB-U	ROC-U	ROD-U	ROE-U	ROF-U	ROG−U	ROH-U	ROI-U

Field	Displacement	Addres	sed by	Modules					
PPIDEPHQ	(280)	ABA-U	ABT-U	ABU-U	AEV-U	ABW-U	ABX-U	ABY-U	ABZ-U
		AGA-M	AGG-U	AGI-U	AGN-M	AOI-M	AOJ-M	AOK-M	AON-N
		AOR-M	AOS-M	APA-U	ADP-U	RBA-U	RBT-U	RBU-U	RBV-U
DD TD = 0 ***	(200)	RBW-U	RBX-U	RBY-U	REZ-U	RGB-U	ROI-U	ROJ-U	ROK-U
PPIDEPHO	(280)	RON-M	ROR-M	ROS-M	ROT-M	RPA-U	RPB-M	RPD-U	RPE-I
	10 11 0 1	8GB-U	80N-M	8PA-M	90N-M				
PPIDEV	(240)	APB-M	RCJ-M	RCK-U	RC1-M	8CK-U			
PP IDEVCD	(452)	AGC-U 9PA-U	AGE-U	APB-U	APE-U	RC 1-M	8CK-U	9GB-U	9GC−t
PPIDIRAD	(118)	AGE-O	RGC-U	ROK-M	ROT-M	RPB-U	RPE-U		
PPIDOOBA	(2E4)	RBL-M	RBM-M	RBN-M	REO-M	RBP-M	ROP-M	ROQ-U	RPG-U
		8BN-U	8BQ-U	9BN-M	9 <b>20-</b> M				
PPIDOUO	(2B4)	API-M	RBM-M	RBO-M	ROP-U	ROQ-U	8BO-M		
PPIDPTRK	(42E)	RCJ-M	ROK-U	ROT-U					
PPIDSKED	(98)	AGB-0	AO 1-M	RCJ-O	RC4-M	ROK-U	8CK-O	80N-M	8PM-C
PPIENDAR	(1A8)	AGE-O	AGH <b>~</b> O	AOK-O	O-TOA	A01-0	APJ-O	RCJ+O	ROK-1
	•	RPB-O	RPC-O	RPE-O	RPF-O	RPM-O	8CK-U	80N-0	8 PM-0
		9GC-0	90N-0						
PPIEOF	(388)	AGA-U	AGI-U	AGN-U	APF-U	RCT-O	RCV-M	RC7-M	RC8-N
	, ,	RGV-U	9GN−U						
PPIFCF01	(77)	<b>AOL</b> ∸U							
PPIFFF	(64)	AOM-U	RC1-M						
PPIFILSZ	(25C)	APB-U	RCJ-U	RCK-M	RC1-M	RFC-M	RPF-M	RPG-U	RPM-N
	<b></b>	8CK-U	8PM-M						
PPIFSEON	(23B)	RPC-U	RPM-U	8PM-U					
PPIFTTAB	(43C)	APB-M	RC 1-M	RPB-U	RPE-U				
PP IGETMN	(30C)	ADE-U	ADG-U	ADS-U	ADT-U	APC-U	APG-M	APH-M	API-N
	(/	APL-M	RCT-U	RCV-U	RPA-U	RPB-U	RPD-U	RPE-U	RPG-U
		8PA-U	9DJ -U						
PP IGETS Z	(310)	APG-M	APH-M	API-M	APL-M	RCT-U	RCV-U	RPA-U	RPB-U
	<b>(</b> - · - <b>/</b>	RPD-U	RPE-U	RPG-U	U-A98				
PPIINSCT	(248)	RDE-M	RDR-M	RDS-M	ROA-U	ROB-U	ROC-U	ROD-U	ROE-U
	<b>V V</b>	ROF-U	ROG-U	ROH-U	ROI-U	ROJ-U	ROK-U	RON-U	80N-U
PPIINT	(378)	AOW-U	AOX-U	AOY-U	AOZ-U	RON-U	80N-U		
PPIIPBLK	(58)	RCK-U	RC1-M						
PPIJOBNM	(430)	RCV-U	RCZ-M	RC1-M					
PPILABO1	(C8)	AGB-M	AGC-M	AGD-M	AGE-M	AGM-M	AOT-O	A05-M	ROT-C
	(33)	RPB-M	RPE-M						
PPILAB02	(2D4)	ADB-U	ADC-U	ADD-U	ADE-U	ADG-U	ADP-U	ADq-U	ADR-U
	• - •	ADS-U	ADT-U	AGA-U	AGB-U	AGC-U	AGD-M	AGE-M	AGF-U
		AGI-U	AGL-M	AGM-M	AGN-U	APF-U	APG-M	APH-M	API-N
		APL-M	RCT-U	RGD-M	RGE-M	RGM-M	8GC-M	9GC-M	9GN-U
PPILAB03	(2AC)	ADB-U	ADC-U	ADD-U	ADE-U	ADG-U	ADH-U	ADL-U	ADP-U
	(/	ADQ-U	ADR-U	ADS-U	ADT-U	AGA-U	AGB-M	AGC-M	AGD-N
		AGE-M	AGF-U	AGI-U	AGL-M	AGM-M	AGN-U	APF-U	APG-N
		APH-M	API-M	APL-M	RCK-U	RCT-U	RC1-M	RGB-U	RGC-U
		RGD-U	RGE-U	RGL-U	RGM-U	8GC-U	9GC-U	9GN-U	
PPILAB04	(2D8)	ABA-U	ABB-U	ABC-U	ABE-U	ABG-U	AGH-U	ABI-U	ABJ-U
	()	ABK-U	ABL-U	ABM-U	ABN-U	ABO-U	ABP-U	ABT-U	ABU-U
		ABV-U	ABW-U	ABX-U	ABY-U	ABZ-U	AGK-U	APA-U	APB-U
		APD-U	APE-U	APF-U	APG-M	APH-M	API-M	APK-U	APL-N
		RPA-U	RPB-U	RPD-U	RPE-U	RPG-U	RPV-U	8PA-U	9BN-U
		9BO-U	9PA-U					J V	2214
PPILAB05	(2DC)	ABL-U	ABM-U	ABN-U	ABO-U	ABP-U	APA-U	APB-U	APD-U
· = =mtzDVJ	(200)	APE-U	APG-M	APH-M	API-M	APL-M	RPA-M	RPB-M	RPD-M
		AFD-U	NEG TH	VE II _ [i]	WLT_M	WETLE.	VEW_M	VEカ_い	ひをわった
			DDC-0	AD A -M	O DM - II	II_OGD	וז – גמס		
PPILAB06	(2E0)	RPE-M AGA-U	RPG-O AGI-U	8PA-M AGK-U	9PN-U AGN-U	9BO-U APF-U	9PA-U APG-M	API-M	APK-U

Field	Displacement	Addres	sed by	Modules			_		
	**************								
PPILAB07	(2B0)	ABA-U	ABB-U	ABC-U	ABE-U	ABG-U	ABH-U	ABI-U	ABJ-U
		ABK-U	ABT-U	ABU-U	ABV-U	ABW-U	ABX-U	ABY-U	ABZ-U
		ADL-U	AGB-U	AGK -U	APA-U	APB-U	APD-U	APE-U	APF-U
		APG-M	APH-M	API-M	APK-U	APL-M	RBA-U	RBE-U	RBI-U
		RBK-U	RBV-U	RBX-U	RCJ-U	RCK-M	RC1-M	RGB-U	RPA-M
		RPB-M	RPD-M	RPE-M	RPG-M	8GB-U	8PA-M	9GB−U	9PA-U
PPILAB08	(2E4)	ADB-U	ADC-U	ADD-U	ADE-U	ADG-U	ADP-U	ADQ-U	ADR-U
		ADS-U RDT-U	ADT-U	APG-M	APL-M	RDP-U	RDQ-U	RDR-U	RDS-U
PPILAB09	(2B4)	APD-U	API-U	APL-U	RCK-M	RC 1-M			
PPILAB10	(308)	AGB-U	APL-M	9GB-U					
PPILEXFD	(2A4)	AOB-U	AOD-U	AOF-U	AOH-U	M-MOA	U-004	300 W	3 D A 11
PPILEXFF	(2A5)	ABF-U	ADB-U	ADC-U	ADD-U	ADE-U	ADG-U	ADP-U	ADQ-U
		ADR-U	ADS-U	ADT-U	AOB-U	AOD-U	AOF-U	AOH-U	AOM-M
DD 77 7117	(0.20)	AOQ-U	RBA-U	RBE-U	D07 11	DO 0 11			
PPILINK	(23C)	RCZ-U	RC 1-M	RC6-U	RC7-U	RC8-U	ADC II	ADR-U	3DC-11
PPIMODEX	(238)	ABQ-U	ABR-U	ABS-U	ADD-U	ADE-U	ADG-U		ADS-U AOB-U
		ADT-U	AGA-U	AGG-U	AGI-U	AGJ-U	AGK-U	AGN-U	
		AOD-U	AOF-U	AOH-U APK-U	APA-U	APB-U	APC-U RCK-U	APF-U RC 1-M	APG-U RC6-U
		APH-U RC <b>7</b> -U	API-U RC8-U	RON-U	APL-U RPC-U	RCH-M RPF-U	RPG-U	RPM-U	8CK-U
			9GN-U	9PA-U	KPC-U	RPF-0	RPG-U	KPM-U	0CV-0
DD TWOUDA	/II E O \	8PM-U	RDC-U	RDD-U	DDO-II	RDR-U			
PPIMOVEQ PPIMRGAL	(458) (294)	ABS-M ADU-M	AOI -U	AOJ-M	RDQ-U AOQ-U	AOR-U	A02-U	APH-U	RCK-M
E E THEGAT	(234)	RC 1-M	ROI-U	ROR-M	ROT-U	AUK-U	AUZ-U	APH-U	NCN-K
PPIMRGMX	(292)	ADH-U	ADI-U	ADJ-M	ADL-U	ADR-U	ADS-U	ADU-M	ADX-U
PPIMRGMA	(292)	AGA-U	AGB-U	AGD-U	ACE-M	AGF-U	AGI-U	AGM-U	AGN-U
		AOI-U	AGD-U AOJ-U	AOD-U	AGE-M	AGF-U	AGI-U	AOT-M	AO2-U
		APF-U	APH-U	API-U	AOK-U	APL-U	RCJ-U	RCK-M	RCL-M
		RCT-U	RC 1-M	RDL-U	RGB-U	RGC-U	RGL-U	ROI-M	ROK-U
		RON-M	ROP-U	ROQ -U	ROR-M	ROS-M	ROI-M	RPC-U	8GB-U
		8GC-U	80N-M	9DJ-M	9GB-U	9GC+U	90N-U	11200	002 0
PPIMRGOP	(296)	ADL-U	AGC-U	AGJ-U	U-QOA	AOR-M	APH-M	ROK-M	ROT-M
	(230)	90N-M							
PPIMXCOL	(450)	RC 1-M	RDE-U	RDG-U	RDS-U	RDI-U	ROP-U	ROQ-U	
PPINDSKA	(2A8)	AGE-U	AGI-U	AGJ-U	AGK-U	AOK-U	U-TOA	A01-U	APB-U
		APE-U	APH-U	RCJ-U	RCK-U	RCT-U	RC1-M	RC4-U	ROK-U
		ROT-U	RPB-U	RPC-U	RPE-U	RPF-U	RPG-U	8CK-U	80N-U
		8PM-U	9GN−U	90N <b>−</b> U					
PPINET	(338)	AOA-U	AOB-U	AOC-U	AOD-U	AOE-U	AOF-U	AOG-U	AOH-U
		AOP-U	U-QOA	U-WOA	AOX-U	AOY-U	AOZ-U	RDB-U	RDC-U
		RDD-U	RDE-U	RDG-U	RDH-U	RDI-U	RDJ-U	RDP-U	RDQ-U
		RDR-U	RDS-U	RDT-U	RDU-U	RDX-U	RGD-U	RGE-U	RGF-U
		RGM-U	ROJ-U	ROW-U	ROX-U	ROY-U	ROZ-U	RPA-U	RPB-U
		RPG-U	8DJ-U	8GC-U	8PA-U	<u></u>			
PPINETAR	(438)	ADH-M	ADI-M	ADJ-M	ADL-M	ADU-M	M-KQA	AGE-M	RDL-U
		RGC-U	ROP-U	ROQ-U	ROT-U	9DJ-M			
PPINETM	(368)	AOP-U	U-QOA	RBG-U	RBH-U	RBI-U	RBJ-U	RBK-U	RBT-U
		reu-u 8gb-u	RBV-U	RBW-U	REX-U	RDL-U	RGB-U	RGC-U	RGL-U
PPINUMCF	(70)	AOL-U	U-MOA	AO5-U	RCZ-U	RC 1-M			
PPIODOM	(120)	AON-O	RON-O						
PPIOPBAD	(454)	ABG-M	ABH-M	ABJ-M	AET-M	ABU-M	ABW-M	RBG-M	RBH-M
		RBI-M	RBJ-M	RBK-M	RET-M	RBU-M	RBV-M	RBW-M	RBX-M
		ROP-U	ROQ-U						
PPIOPBLK	(272)	AGK-U	APF-U	APK-U	RC1-M				
PPIOPEN	(3B0)	AGA-U	AGB-U	AGC-U	AGD-U	AGE-U	AGG-U	AGH-U	AGI-U
		AGJ-U	AGK-U	AGL-U	AGM-U	AGN-U	APA-U	APB-U	APC-U
		APD-U	APE-U	APJ-U	APK-U	RCI-O	RPA-U	9GB−U	9GC−U
		9GN−U	9PA-U						

Field	Displacement	Addres	ssed by	Modules					
nn vn	<i>(</i> (0)		204 15						
PPIPBUFF	(68)	AOM-U	RC 1-M						
PPIPCF01	(72)	AOL-O	AOM -O	357.0	3.011.0	304.0	DD I . A	DOX-14	non. I
PPIPDWA	(48)	ADH-O RPQ-U	ADI -O 9DJ -O	ADJ-O	ADU-O	ADX-O	RDL-0	ROK-M	ROP-U
PPIPGCSZ	(60)	RCB-U	RCK-U	RC1-M					
PPIPSVA	(88)	ABQ-M	ABR-M	ABS-M	ADJ-M	AGB-M	AGL-M	APC-M	API-N
	(	RGB-O	RGD-O	RGF-U	RGL-O	RGM-O	8GB-O	8GC-O	9DJ-N
PPIP1ASZ	(4C)	RC 1-M							
PPIP1GC	(48)	APG-U	APL-U	RCK-M	RC1-M				
PPIP2GC	(2B8)	APH-U	RCK-M	RC1-M					
PPIP3ASZ	(2C0)	RCV-U	RC 1-M						
PPIP3GC	(2BC)	API-U	RCK-M	RC1-M					
PPIRCDCT	(24C)	ADH-U	ADI-I	AGD-U	AGH-U	AGM-U	APK-U	RPC-M	RPF-M
		RPG-U	RPM-M	8PM-M	9GC−U				
PP IRCDL 1	(288)	ADB-U	ADP-U	AGA-U	AGI-U	AGN-U	APF-U	APG-U	API-U
DD 200 00	(0.00)	APL-U	RCK-U	RC1-M	RDE-U	RDS-U	9GN−U		
PPIRCLD2	(28A)	ABB-U	ABC-U	ABG-U	APH-U	ABJ-U	ABR-U	ABS-U	ABT-U
		ABU-U	ABW-U	ABY-U	ABZ-U	ADH-U	ADL-U	ADX-U	AGD-U
		AGE-U	AGM-U	APB-U	APE-U	RBB-U	RBC-U	RBG+U	RBH-U
		RBJ-U RC6-U	RBT-U	RBU-U RDG-U	REW-U	RBY-U RDL-U	RBZ-U RDS-U	RCJ-U	RCK-U
		RGB-U	RDE-U RGC-U	RGE-U	RDH-U RGL-U	RGM-U	8CK-U	RDT-U 8GB-U	RDX-U
PPIRCDL3	(28C)	ABL-U	ABQ-U	AGK-U	APF-U	API-U	APK-U	RCK-U	RC8÷U
PPIRCDL4	(28E)	RCJ-U	ROI-U	ROJ-U	ROK-U	RON-U	ROR-U	RPE-U	80N-U
PP IRCDL5	(290)	APG-U	RCJ-U	RCK-U	ROS-U	8CK-U	1011 0	KEE- O	OOM
PPIRCV	(230)	AGA-U	AGB-U	AGD-U	AGE-M	AGF-U	AGH-U	AGI-U	AGK-U
		AGM-M	AGN-U	U-MOA	APC-U	APF-U	APJ-U	APK-U	RCV-M
		RC9-U	RGB-M	RGC-M	RGD-M	RGE-M	RGL-M	RGM-M	ROA-U
		ROB-U	ROC-U	ROD-U	ROE-U	ROF-U	ROG-U	ROH-U	ROI-U
		ROJ-U	ROK-U	RON-U	ROR-U	ROS-U	RPA-M	RPB-M	RPD-M
		RPE-M	RPG-U	8GB-M	8GC-M	8PA-M	8PM-U	9GN-U	9PA-U
PP IRD	(358)	AGB-U	AGC-U	AGD-U	AGE-U	AGL-U	AGM-U	RBG-U	RBH-U
	•	RBI-U	RBJ-U	RBK-U	RBT-U	RBU-U	RBV-U	RBW-U	RBX-U
		RCV~U	RDH-U	RDI-U	RDX-U	RON-U	ROR-U	ROS-U	ROT-U
		80N-U	9BG−U	9GC−U					
PP IRECFM	(276)	AGK-U	APF-U	APK-U	RC1-M				
PP IRMA	(390)	AGA-U	AGI-U	AGN-U	AOJ-U	RDD-U	RDE-U	RDG-U	RDR-U
		RDS-U	RDT-U	RGV-U	ROI-U	ROJ-U	ROK-U	ron-u	RPC-U
		80N-U	9GN-U						
PPIRMB	(398)	RBJ-U	RBK-U			ROP-U	ROQ-U	ROR-U	ROS-U
DD TDMC	(200)	RPE-U	RPF-U	RPM-U	8PM-U	505 ti	700 0	220.22	
PPIRMC	(398)	RBM-U	RBO-U	RDU-U	RGV-U	ROP-Ü	ROQ-U	RPG÷U	RPV-U
PPIRPLDP	(45C)	8BO-U	ACT -M	AGK-M	2 C21 M	3 DTD_M	3 DVW	DDN_E	nno. H
FEIKETOR	(450)	AGA-M RGV-U	AGI-M RPG-U	RPV-U	AGN-M 9GN-M	APF-M	APK-M	RBN-U	RBO-U
PPISBLCT	(2FC)	ADH-U	ADI-U	ADX-U	AGB-M	AGC-M	AGD-M	ACE_M	AGL-M
F F TODUC I	(250)	AGM-M	RBG-U	RBH-U	REI-U	RBJ-U	RBK-U	AGE-M RBT-U	RBU-U
		RBV-U	RBW-U	RBX-U	RDH-U	RDI-U	RDL-U	RDX-U	RGB-U
		RGC-U	RGD-U	RGE-U	RGL-U	RGM-U	8GB-U	8GC-U	9GB-M
		9GC-M							, H
PPISEQCT	(250)	AGE-O ROT-O	AGK-O	AOR-M	AOT-O	APH-O	ROI-M	ROK+0	ROR-M
PPISESDS		AGK-U	APF-U	APK-U					
PPISIRG	(264)	RC 1-M	RPA-U						
PPISKPRD	(50)	APC-U	RC 1-M						
	•	AGK-U	APF-U	APK-U					
PPISKSDS	•								
PPISKSDS PPISLIB PPISORCE	(31C) (314)	RCV-U RCZ-M	RCZ-M RC1-M	RC6-U RC9-U	RC7-U	RC8-U	RC9-U		

Field	Displacement	Addres	sed by	Modules					
************									
PPISPGN 1	(2D0)	ABQ-M	ABR-M	ABS-M	ABT-M	ABU-M	ABV-M	ABW-M	ABX-M
		ADH-M	ADI-M	ADJ-M	ADL-M	ADU-M	ADX-M	AGA-M	AGB-M
		AGC-M	AGD-M	AGE-M	AGG-M	AGI-M	AGJ-M	AGK-M	AGL-M
		AGM-M	AGN-M	AOA-M	AOB-M	AOC-M	AOD-M	AOE-M	AOF-M
		AOG-M	M-HOA	AON-M	M-QOA	APA-M	APB-M	APD-M	APE-M
		APF-M	APG-M	APH-M	API-M	APK-M	API-M	RCT-U	9DJ-M
	.0.0 % .	9GB-M	9GC-M	9GN -M	90N-M	9PA-M			H
PPISRALG	(264)	AGE-U RPE-M	AGI-M 8GB-U	AGJ-M 8GC-U	AGK-M 8PA-U	RGC-U 9GN-M	RGE-U	ROT-M	RPB-U
PPISRTEL	(270)	ABB-U	ABC-U	ABG-U	ABH-U	ABJ-U	ABT-U	ABU-U	ABW-U
	(2.0)	ABY-U	ABZ-U	ADH-U	ADL-U	AGD-U	AGE-U	AGM-U	APB-U
		APE-U	APG-U	RBB-U	RBC-U	RBG-U	RBH-U	RBJ-U	RBT-U
		RBU-U	RBW-U	RBY-U	REZ-U	RCJ-U	RCK-M	RC1-M	RGB-U
		RGD-U	RGL-U	RGM-U	ROI-U	ROJ-U	ROK-U	RON-U	ROR-U
		ROS-U	RPB-U	RPE-U	8GB-U	8GC-U	80N-U	9GC−Ū	-
PPISRTG	(26C)	AOA-U	AOB-U	AOC-U	AOD-U	AOE-U	AOF-U	AOG-U	AOH-U
	<b>( ,</b>	APG-M	APL-M	RCJ-U	RCK-M	RC1-M	ROI-U	ROJ-U	ROK-U
		ROW-U	ROX-U	ROY-U	ROZ-U				
PPISTAR	(120)	AGI-O	AGJ-O	AGK-O	AOK-O	AOT-O	A01-0	APB-O	APE-O
		RCJ-O	8CK-O	9GN -0	90N-0				
PPISTDCB	(2F8)	ABL-U	ABM-U	ABN-U	ABO-U	ABP-U	ADB-U	ADC-U	ADD-U
		ADE-U	ADG-U	ADJ-U	ADP-U	ADQ-U	ADR-U	ADS-U	ADT-U
		AGA-M	AGB-U	AGD-U	AGE-U	AGF-U	AGG-M	AGH-U	AGI-M
		agj-m	AGK-M	AGL-U	AGM-U	AGN-M	APA-U	APB-U	APC-U
		APD-U	APE-U	APF-M	APJ-U	APK-M	RCT-U	RGB-U	RGC-U
		RGD-U	RGL-U	RGM-U	RON-U	ROS-U	RPA-U	RPB-U	RPC-U
		RPD-U	RPE-U	RPF-U	RPG-U	RPM-U	8GB-U	8GC−Ü	8PA-U
		8PM-U	9BN-U	9BO-U	9DJ-U	9GB−U	9GC−U	9GN-M	
PPISTIOB	(300)	AGA-M	AGC-U	AGE-U	AGI-M	AGJ-M	AGK-M	APB-U	APD-U
		APE-U	RGC-U	RGE-U	RPB-U	RPE-U	8GC−U	9GC−U	9GN-M
DDTCTDNM	(448)	9PA-U RCV-U	RCZ-M						
PPISTPNM PPISVARE		AGD-M	AGM-M						
PPISVARE	(0)	AGD-M AGA-U	AGM-M AGF-U	AGI-U	AGN-U	APC-U	APF-U	APG-U	APL-U
LLTOADTM		RCT-U	RC6-U	RC8-U	RGF-U	9GN-U	APP-0	AFG-0	WLT1.0
PPISVSOP		AGK-U	APF-U	APK-U	RBN-U	RBO-U	RCT-U	RC8-U	RPG-U
PPISW1	(230)	ABA-U	ABB-U	ABC-U	ABE-U	ABF-U	ABG-U	ABH-U	ABI-U
I I IDW	(250)	ABJ-U	ABK-U	ABL-U	ABM-U	ABN-U	ABO-U	ABP-U	ABQ-U
		ABR-U	ABS-U	ABT-U	ABU-U	ABV-U	ABW-U	ABX-U	ABY-U
		ABZ-U	ADH-U	ADI -U	ADJ-M	ADL-U	ADP-M	ADQ-M	ADR-M
		ADS-M	ADT-M	ADU-M	AGA-U	AGB-M	AGC-M	AGD-M	AGE-M
		AGG-U	AGH-U	AGI-U	AGJ-U	AGK-U	AGI-M	AGM-M	AGN-U
		AOA-U	AOB-U	AOC-U	AOD-U	AOE-U	AOF-U	AOG-U	AOH U
		AOL-U	AON-U	AOP-U	U-QOA	AOR-M	AOS-M	AOT-U	A01-U
		AO2-U	APA-U	APB-M	APC-U	APD-U	APE-U	APF-U	APG-M
		APH-M	API-M	APJ-U	APK-U	APL-M	RBM-M	RBO-U	RCB-U
		RCH-M	RCJ-U	RCK-M	RCT-U	RCV-M	RC Z-M	RC1-M	RC4+U
		RC6-U	RC7-U	RC8-M	RC9-U	RDD-M	RDE-M	RDL-U	RDP-M
		RDQ-M	RDR-M	RDS-M	RDT-M	RDU-U	RGB-M	RGC-M	RGD-M
		RGE-M	RGF-U	RGL-M	RGM-M	RGV-U	ROI-M	ROJ-M	ROK-U
		RON-M	ROP-U	ROQ-U	ROR-M	ROS-M	ROT-M	ROW-U	ROY-U
		RPA-M	RPB-M	RPC-M	RPD-M	RPE-M	RPF-M	RPG-U	RPM-M
		8BO-U	8CK-U	9GB-M	8GC-M	80N-M	8PA-M	8PM-M	9BN-U
		9BO-U	9DJ-M	9GB-M	9GC-M	9GN-U	9PA-U		
PPISW14		AGC-U	AGE-U	AGI-U	AGJ-U	AGK-U	APB-U	APE-U	RGC-U
		RGE-U	RPB-U	RPE-U	8GB-U	8GC−U	9GB-U	9GC+U	9GN-U
		9PA-U							:

Field	Displacement	Addres	sed by	Modules					
		•							
PPISW2	(23B)	ABS-U	AGA-U	AGF-U	AGI-U	AGK-U	AGN-U	NOM-U	A05-1
		APC-U	APF-U	APG-U	APK-U	APL-U	RBL-U	RBM-U	RBN-U
		RBO-U	RBP-U	RCB-U	RCK-U	RCI-U	RC Z-M	RC1-M	RC6-I
		RC8-U	RDB-U	RDC-U	RDD-U	RDE-U	RDG-U	RDI-U	RDP-U
		RDQ-U	RDR-U	RDS-U	RDT-U	RGF-U	RPC-U	RPG-U	RPM-
		8BN-U	8B0-U	8PM-U	9GN-U				
PPITAVLC	(2C8)	APG-U	APL-U	RCK-M	RC1-M				
PPITPCYL	(42C)	AGC-U	AGE-U	AOK -U	APB→U	APE-U	RCJ-M	ROT-U	8CK-
		90n-U							
PPITPPT	(F4)	AOR-M	AO 2-U	APK-U	RC4-M	ROI-M			
PPITPTBL	(F8)	AGA-O	AGB-0	AGD-O	AGG-O	AGH-O	AGI-O	AGM-O	AGN-0
		AOI-O	AOJ-O	AON-O	AOR-O	AOS-O	A02-0	APJ-O	APK-0
		RGB-O	RGL-O	ROI-O	ROJ-M	RON-O	RPC-0	PPF-0	RPM-U
PPIUNTCT	(304)	ABT-M	ABU-M	ABV-M	APW-M	ABX-M	RBA-U	RBT-U	RBU-I
		RBV-U	RBW-U	RBX-U	REY-U	RBZ-U	RON-U	80N-U	
PPIVMV	(350)	ABF-U							
PPIVSI	(464)	RC6-M							
PPIVSMSG	(460)	AGA-M	AGI-M	AGK-M	agn-m	APF-M	APK-M	RCT-U	RGV-1
		RPV-U	9GN-M						
PPIVSMSL	(45E)	AGA-M	agi-m	AGK-M	AGN-M	APF-M	APK-M	RCT-M	RGV-1
		RPV-U	9GN−M						
PPIWKARE	(48)	ABF-M	AGD-O	AGF-M	AGL-O	AGM-O	M-TOA	APB-O	APE-0
		RCT-U	RCV-M	RGB-M	RGC-M	RGD-M	RGE-M	PGL-M	RGM-1
		RGV-M	ROI-M	ROJ-M	ROK-M	RON-M	ROR-M	ROS-M	ROT-1
		RPB-M	RPC-M	RPE-M	RPF-M	RPG-M	RPM-W	RPV-M	8GB-1
		M-NO8	8PA-M	8PM-M	9GC-O	• .			
PPIWRT	(348)	ABC-U	APA-U	APB-U	APD-U	APE-U	RBA-U	RBB-U	RBC+
		RBE-U	RBG-U	RBH-U	REI-U	RBJ-U	RBK-U	RBL-U	RBM-U
		RBN-U	RBO-U	RBP-U	RBT-U	RBU-U	RBV-U	RBW-U	PBX-U
		RBY-U	RBZ-U	RC9-U	ROI-U	ROK-U	ROR-U	ROT-U	8BN-1
		U-088	80N-U	9PA-U					
PPIXCAP	(268)	APD-U	ADQ-U	ADR-U	ADS-U	ADT-U	AON-U	RCJ-M	RC1-1
		RDD−U	RDE-U	RDP-M	RDQ-M	RDR-M	RDS-M	RDT-M	ROK-1
		RON-M	RPA-M	RPB-M	8PA-M				
PPIX11	(3B8)	APG-U	APL-U	EX1-0	RC6-M	RC7-0	RC9-0		
PPIX15	(3C8)	ADD-M	ADE-M	ADR-M	ADS-M	ADU-M	RC6÷0	RC8-M	RDD-U
		RDE-U	RDR-U	RDS-U	RDU-U				
PPIX16	(4 10)	RCT-0	RC6-M	RDD-U	RDE-U	RDP-U	RDS-U		
PPIX17	(3D8)	RC 6-0	RPC-U	RPM-U	8PM-U				
PPIX18	(3E8)	AGA-U AGA-U	AGI-U	AGN-U	RC6-O	9GN−U			
PPIX19	(3F8)	AGA-U	AGI-U	AGN-U	RC6-O	9GN−U			
PPIX21	(3C0)	APH-U	APL-U	EX2-O	RC7-M				
PPIX25	(3D0)	RBJ-U	RBK-U	RBW-U	REX-U	RC 7-0			
PPIX27	(3E0)	RC 7-0	RPF-U	RPM-U	8PM-U				
PPIX28	(3F0)	AGG-U	AGJ-U	AGN-U	RC7-0	9GN−U			
PPIX29	(400)	AGG-U	AGJ-U	AGN-U	RC7-0	9GN−U			
PPIX31	(3C0)	API-U	EX3-0	RC8-M					
PPIX35	(3D0)	ABM-M	ABO-M	RBM-U	REO-U	RC8-0	8BO-U	9BO-M	
PPIX37	(3E0)	RC8-0	RPG-U						
PPIX38	(3F0)	AGK-U	APF-U	APK-U	RC8-O				
PPIX39	(400)	AGK-U	APF-U	APK-U	RC8-0				
PPIX61	(408)	AOB-U	AOD-U	AOF-U	AOH-U	U-QOA	RC6-0	RC7-0	RC8-0

## COMMON CROSS-REFERENCE TABLE (BLOCKSET)

The following table contains the names of COMMON fields together with their displacements.

	•	with their dis	pracements.
Disp.	<u>Name</u>	<u>Definition</u>	Description
00EC 0388	COMABEND COMACE	DS C DS F	ABDUMP DD ENCOUNTERED Abnornal Channel End
05FD 0250 01A8 0142	COMALTAB COMAREAQ COMARECS COMARKER	DS CL256 DS F DS F DS C	WORK AREA PREFERENCE QUEUE ADDED REC COUNT CONTROL CARD ERROR POSITION FLAG
01E4 0830 0198	COMAVAIL COMAVGLN COMAXIO	DS F DS F DS H	ADDRESS OF AVAILABLE COMMON AVERAGE RECORD LENGTH VALUE MAXIMUM SORTIO BUF TRANSFER COUNT
03A4 0118 090C	COMAXKPB COMAXLIM COMBACK	DS H DS F DS F	MAXIMUM KEYS PER BLOCK MAX FOR SIZE=MAX BACKOUT RECORD'S PREDECESSOR
0254 01C0 0264	COMBINEQ COMBINSN COMBINSQ	DS F DS F	QUEUE OF AVAILABLE RECORD BINS SELECTED BIN RECORD COUNT SELECTED BIN QUEUE
0268 02AC 082A	COMBINST COMBINSZ COMBLKS	DS F DS F DS C	SELECTED QUEUE TAIL SIZE OF A RECORD BIN N=BYPASS FLR BLOCKSET
020C 041C 0418 0390	COMBLKSI COMBLOKI COMBLOKO COMBOUND	DS F DS F DS F	WORKFILE BLOCK SIZE INPUT BLOCK COUNTER OUTPUT BLOCK COUNTER COUNT OF WITHIN BOUND RECORDS
0384 0374 027C	COMBUCTR COMBUD COMBUDAQ	DS H DS F DS F	HIGH BV XFER COUNT CURRENT BVD ALLOCATED BVD COUNT
0278 0284 0280	COMBVDFQ COMBVDHI COMBVDLO	DS F DS F DS F	FREE BVD QUEUE ADDRESS OF HIGH BOUNDARY VALUE DEF ADDRESS OF LOW BOUNDARY VALUE DEF
03B6 0080 0274	COMBVLIM COMB37R COMCADAQ	DS H EQU X'80' DS F	HIGH BV XFER LIMIT B37 INFO MESSAGE FLAG MASK AVAILABLE CYLINDER AREA QUEUE
0270 02CC 08EC	COMCADFQ COMCCHH COMCENT	DS F DS F DS F	FREE CAD QUEUE LATEST SORTIO CCHH PER CENT RSA UTILIZATION
081D 081E	COMCHALT	DS CL1	A=TRANSLATE AQ ONLY C=TRANSLATE CH AND AQ Y=RECORD COUNTERS CHECKED
0484 082B	COMCLOSE COMCNTL	DS OF DS C	N=RECORD COUNTERS NOT CHECKED CLOSE LIST AREA X'00'=SORTCNTL NOT PRESENT
00E8 00E9 0139	COMCOBIN COMCOBUT COMCPRNT	DS C DS C	COBOL INPUT MODE OPTION COBOL OUTPUT MODE OPTION COBOL SORT MESSAGE PRINT CLASS
0130 03B0 08D4	COMCSYSO COMCYLNO COMDELTA	DS CL8 DS H DS F	COBOL SYSOUT=A DDNAME LOGICAL CYLINDER ASSIGNMENT CTR RESIDUE DELTA
00F8 01AC 0358 0248	COMDIAGS COMDRECS COMDUBBL COMDXKEY	DS F DS F DS D DS F	DIAGNOSTIC LIST ADDRESS DELETED REC COUNT DOUBLEWORD ALIGNED WORK AREA ADDRESS OF KEY AREA DXL
023C 0244 0238	COMDXLIO COMDXLRQ COMDXLSQ	DS F DS F DS F	SORTIO CIRCULAR LIST ENTRY SORTWK DXL READ QUEUE SEQUENTIAL DXL LIST HEAD
02B4 0240 024C	COMDXLSZ COMDXLWQ CONDXSKY	DS F DS F DS F	SIZE OF DXL (DEPENDS ON V OR R) SORTWK DXL WRITE QUEUE SECONDARY KEY AREA DXL ADDRESS
05F0 0814 05F8	COMDYNAM COMDYNER COMDYNUM	DS CL8 DS F DS CL8	DYNALLOC UNIT NAME RETURN CODE FROM DYNALLOC DYNALLOC COUNT
092C 0128	COMECTRS COMEMBRI	DS 10F DS CL8	EVENT COUNT SAVE AREA Sysout file name

# Licensed Material--Property of IBM Page of LY33-8042-6, As Updated 31 March 1981, By TNL LN20-9345

	Disp.	Name	<u>Definition</u>	Description
١	0130 00F2	COMEMBRO COMEOF	DS CL8 DS C	COBOL SYSOUT=A DDNAME END OF FILE ON SORTIN INDICATOR
1	00F1 02A4	COMEOU	DS C DS F	END OF VOLUME ON SORTIN INDICATOR EQUAL COUNTER
ı	0484	COMEQUAL COMERGEI	DS 28F	CAONE COOKIEK
ı	04F4	COMERGES	DS 48F	FUENT COUNT CAME ADDA
J	0920 02A8	COMESAVE COMEXITS	DS 3F DS F	EVENT COUNT SAVE AREA EXIT LIST HEAD
ı	0960	COMEXTRA	DS 2F	UNUSED WORDS
ı	00E4	COMFAULT	DS OF	END OF FOUNDATION
ı	0968 01D4	COMFEND COMFILEG	DS 0D DS F	GET FILE AREA PTR
1	01D8	COMFILEP	DS F	PUT FILE AREA PTR
١	019C 0010	COMFILEZ COMFLAG	DS F EQU 16	FILE SIZE TRACE INPUT BLOCKS
ı	08D0	COMFLOOR	DS F	PAGE SIZE COMPLEMENT
ı	00E4	COMFOUND	DS OF	CDEE DIN COUNT
1	01BC 08E8	COMFREEB COMFSTOR	DS F DS F	FREE BIN COUNT FIXED RESTORE ROUTINE ADDRESS
١	0420	COMHASH	DS F	SUM OF OUTPUT COUNT IDS
1	0145	COMIBEND	DS C DS H	ABEND IF ERROR DURING INITIAL PHASE INPUT FILE AVERAGE BLOCK LENGTH
Į	0192 018A	COMIBLKL COMIBLKZ	DS H DS H	INPUT BLOCKSIZE
ı	036C	COMIDKEY	DS F	MIDDLE KEY
ı	0484 0146	COMIDSHM COMIDUMP	DS C	INPUT DATA SET NAME DUMP OPTION FOR IBEND
ı	0188	COMIFORM	DS C	INPUT REDFM
ı	0380	COMIIOB	DS F	INPUT IOB
ı	01F4 0154	COMINCOR COMINDOB	DS F DS F	MINIMUM SPACE NEEDED TO SORT SORTIN DCB ADDRESS
ı	0918	COMINFIL	DS F	SORTIN BYTE COUNT
J	01F0 0196	COMINGET COMINIO	DS F DS H	MINIMUM GETMAIN AMOUNT MINIMUM SORTIO BUF TRANSFER COUNT
ı	03A2	COMINKPB	DS H	MINIMUM KEYS PER BLOCK
1	OOEF	COMINDUT	DS C	COMMON IN/OUT DATA SET
ı	08D8 00FC	COMINSEG COMINUS4	DS F DS F	MIN SIZE FOR 1ST SEGMENT MINUS 4
1	0100	COMINUS8	DS F	MINUS 8
ı	0260 0230	COMIOBXQ	DS F DS F	QUEUE OF EXCP IOBS SORTIO DATA ADDRESS IN IOX
ı	02BC	COMIORPB	DS F	SORTIO REC PER BLOCK
J	00F0	COMICTYP	DS C	V=VIRTUAL, R=REAL
ı	0224 0228	COMIOXAD COMIOXSZ	DS F DS F	ADDRESS OF IOX LENGTH OF IOX
l	01A0	COMIRECS	DS F	INPUT REC COUNT
1	018E 08C0	COMIRECZ	DS H	INPUT LRECL
ł	01CC	COMIRES COMIRSAZ	DS F DS F	START OF RESIDUAL INPUT DATA INPUT RSA SIZE
ı	0370	COMISCEL	DS 4C	SWITCH VALUES
Į	0001 0143	COMISREL COMJOB	EQU 1 DS C	O => RELEASE SORTWORK SPACE TYPE/LOG JOB NAME
ı	0220	COMKBLKZ	DS F	SIZE OF KAD BLOCK ENTRY
1	021C 03AA	COMKDATA COMKDELT	DS F	SIZE OF KAD PREFIX
ı	026C	COMKEYFQ	DS H DS F	PRIMARY KEY MULTIPLE LOAD DELTA FREE KEY HOLDER QUEUE
	0398	COMKEYPC	DS H	PRIMARY KEY COUNT
l	039C 0288	COMKEYPN COMKEYPQ	DS H DS F	PRIMARY KEY MINIMUM PRIMARY KEY QUEUE
	028C	COMKEYPT	DS F	PRIMARY KEY QUEUE TAIL
	039E 039A	COMKEYPX COMKEYSC	DS H DS H	PRIMARY KEY MAXIMUM SECONDARY KEY COUNT
	0290	COMKEYSQ	DS F	SECONDARY KEY QUEUE
1	03A0	COMKEYSX	DS H	SECONDARY KEY MAXIMUM
	0294 0298	COMKIRRF COMKIRRQ	DS F DS F	KAD IRRR HOLDER FREE QUEUE KAD IRRR HOLDER QUEUE
ı				

Disp.	Name	<u>Definition</u>	Description
0174	COMKLOAD	DS F DS F	KEY LOAD SUBROUTINE ADDRESS
01D0 0214	COMKOUNT COMKPB	DS F DS F	PRIMARY KAD COUNT KEYS PER WORKFILE BLOCK
0218	COMKPFIX	DS F	SIZE OF BIN KAD PREFIX
03A6	COMKRECZ	DS H	KEY RECORD LENGTH
0954	COMKSAVE	DS F	KEY SAVE AREA ADDRESS
0178	COMKUNLD	DS F	KEY UNLOAD SUBROUTINE ADDRESS
080C 0258	COMLADAD COMLADFQ	DS F DS F	BLOCKED LAD ADDR/ADDR (E35) LAD AVAILABLE LIST
037C	COMLADLM	DS F	LAD AREA LIMIT
025C	COMLADNO	DS F	NEXT STRING LAD QUEUE
0378	COMLADS	DS F	LAD AREA ADDRESS
013C 0180	COMLCTR COMLDKAD	DS C DS F	INITIAL LINE COUNTER VALUE ADDRESS OF KAD LOAD ROUTINE
08F8	COMLEND	DS F	UNBALANCED LIST END
0394	COMLIMIT	DS F	BOUND LIMIT
013D	COMLINES	DS C	LINES PER PAGE
08F4 01FC	COMLIST COMLISTA	DS F DS F	UNBALANCED LIST HEAD LIST HEADER FOR MERGE SORT
0200	COMLISTE	DS F	LIST HEADER FOR MERGE SORT
016C	COMLOAD	DS F	LOAD SUBROUTINE ADDRESS
042C	COMLOCAL	DS 22F	
013B	COMLOG	DS C DS F	LOG MESSAGE CLASS
0900 022C	COMLOKEY COMLOWIO	DS F DS F	POINTER TO LOW LIMIT KEY AREA LOW I/O ADDRESS
0149	COMLPA	DS C	IPUT AND OPUT ARE IN LPA
095C	COMLSTAB	DS F	LENGTH STATISTICS TABLE
03AC	COMLSTAR	DS H	1ST TWO BYTES OF JECLSTAR
0958 011C	COMLSTAT COMMAIN	DS F DS F	LENGTH STATISTICS CTR PTR MAX FOR SIZE= NOT SPECIFIED
OIBC	COMMAINO	DS OF	SAVED INIT MAIN STORAGE VALUE
0100	COMMAIN1	DS F	SELECTED BIN RECORD COUNT
0108	COMMAIN2	DS OF	USED MAIN STORAGE VALUE
01DC 01E0	COMMAREA COMMEND	DS F DS F	ADDRESS OF COMMON WORK AREA ADDRESS OF END OF COMMON WORK AREA
OOEB	COMMNOGO	DS C	ERROR ENCOUNTERED INDICATOR
0000	COMMON	DS 3F	START OF OS SAVE AREA
08BC	COMMSGPR	DS H	MESSAGE PARAMETER SAVE AREA
0828 0020	COMMSGSW Commsnap	DS X EQU 32	INFORMATION MSG FLAG BYTE SNAP WRITE BACK BLOCKS
ÖÖEÉ	COMMTEST	DS C	TEST STATUS INDICATOR
0008	COMMTPHO	EQU 8	ISSUE PHASE O MESSAGES
0040	COMMISEQ	EQU 64	SEQUENCE CHECK SORTOUT
0020	COMMTVER COMMT190	EQU 32 EQU 128	VERIFY BIN COUNT ISSUE 190 MESSAGES
0968	COMMVARY	EQU *	BEGINNING OF VARIABLE AREA
0820	COMMXBLK	DS F	MAX SORTIN BLKSIZE FOR SMF
0104	COMMXCTL	DS CL8	NAME TO XCTL TO
0428 0147	COMNMAX COMOBEND	DS F DS C	ESTIMATED NMAX VALUE
018C	COMOBLKZ	DS C DS H	ABEND IF ERROR DURING OTHER PHASE OUTPUT BLOCKSIZE
0148	COMODUMP	DS C	DUMP OPTION FOR OBEND
0189	COMOFORM	DS C	OUTPUT RECFM
0384	COMOIOB COMOJECB	DS F DS F	OUTPUT IOB ADDRESS OF SORTOUT JFCB
0364	COMOLEND	DS F	END OF OPEN LIST
0360	COMOLIST	DS F	OPEN LIST ADDRESS
0834	COMOPLOC	DS 20F	OPTION STATEMENT WORK AREA
0104	COMOPTNS COMOPTZ	DS OF EQU *-com(	OPTHS SIZE OF OPTIONS
0100	COMOPWTO	DS CL12	WTO HEADER AND ROUTINE
01A4	COMORECS	DS F	OUTPUT REC COUNT .
0190	COMORECZ	DS H	OUTPUT LRECL
08C8	COMORES	DS F	START OF RESIDUAL OUTPUT DATA

<u>Disp.</u> Nam	ne <u>Defir</u>	ition	Description
0400 COMC 03BE COMC 017C COMC 03C4 COMF 03F4 COMF 03FA COMF 03E8 COMF 02B8 COMF 0908 COMF 0908 COMF 03D0 COMF	PALTA DS PALTB DS PARE DS PAREZ DS PARZ6 DS PCOPA DS PCOPB DS PCOPC DS	F CL18 CL6 F CL6 F F CL6 CL6 CL6 CL6 CL6	ALTERNATE REC MOVE MVC 0(KADLIST,R1),0(R15) MVC 0(,R1),0(R15) ADDRESS OF RECORD MOVE ROUTINE CLC 8(,R4),0(R3) CLC 4(,R2),0(R4) CLC 4(,R3),0(R4) CLC 0(,R14),0(R15) LENGTH OF COMPARE OFFSET IN BIN TO SEG BYTE CLC 4(,R4),0(R14) CLC 4(,R5),0(R14) CLC 0(,R4),0(R14)
013F COMF	PEERS DS PFIX DS PICK DS PLADS DS	F CL3 CL6 6F	PEERAGE ACCUMULATOR PREFIX NAME CLC 0(,R2),BVDKEY-BVDSECT(R4)
07EC COMF 0774 COMF 07F0 COMF	PLAST DS PLIST DS PLONG DS PMODE DS	9 30 6 6 C	LAST LIST ENTRY 'L'=LOW, 'H'=HIGH
03B8 COMF 03DC COMF 014C COMF	PMOVE DS PPOUND DS PRDCB DS PRDEC DS	OH CL6 F F	CLC O(,R2),O(R8) ADDRESS OF SYSOUT DCB ADDRESS OF SYSOUT DEC
0138 COMF 03B2 COMF 03EE COMF	PRINT DS PRIZE DS PSLAB DS PSLAK DS	C H CL8 CL6	PRINT MESSAGE CLASS PRIOR BLOCK SIZE CLC 4(,R4),4(R5) CLC 4(,R4),0(R14)
03D0 COMF 0150 COMF 0168 COMF	SLBK DS RDDCB DS RDDEC DS RECFM DS	CL6 F F C	CLC 4(,R5),O(R14) ADDRESS OF SYSIN DCB SYSIN DECB ADDRESS SORT RECORD FORM
02D0 COME 0210 COME 01C8 COME	RECHO DS	2C F F F	LATEST R RECORDS PER WORKFILE BLOCK RSA SIZE REGISTER SAVE AREA
0888 COMF 08E0 COMF 08E4 COMF	RSAV2 DS RSAWK DS RSTOR DS RTAIL DS	F F F	REGISTER AREA REGISTER AREA RSA TO WORK BUF ROUTINE RESTORE LIST POINTER LAST FOUR BYTES OF RECORD
01C4 COME 01B0 COME 01E8 COMS	RUNS DS RUNSZ DS	F F F F	RUNS DISTRIBUTED BY INPUT PHASE CURRENT RUN LENGTH ADDRESS OF FIRST SOTRAGE AREA DESC TIOT ENTRY FOR SORTIN
02C8 COMS 000C COMS 0048 COMS	SAVEO DS SAVEO DS SAVE1 DS SAVE2 DS	F 15F 13F 13F	TIOT ENTRY FOR SORTOUT LEVEL ZERO SAVE AREA LEVEL ONE SAVE AREA LEVEL TWO SAVE AREA
0120 COMS 013E COMS 0910 COMS	SAVE3 DS SERVE DS SGLST DS SGTOT DS	F	LEVEL THREE SAVE AREA AMOUNT OF RESERVED MEMORY LIST CONTROL CARDS OPTION TOTAL BYTES SEGMENTED
038C COMS 00F3 COMS	SHORT DS SIRRR DS SKEYS DS SKIPM DS SMF DS	F C C C	MAIN STORAGE DEFICIT SECONDARY KEYFILE IRRR SECONDARY KEEP PRODUCED 'Y'=SKIP INTERMEDIATE MERGE 'S'=SHORT SMF RECORD 'F'=FULL SMF RECORD (NOT VALID FOR FLR)
0124 COMS 0368 COMS 0144 COMS		CL4 F C	'N'=NO SMF RECORD 'SORT' DD PREFIX OR ALIAS SQUARE ROOT OF RPB TYPE/LOG STEP NAME

Disp.	Name	<u>Definition</u>	Description
01EC 0184 014A 0818 00ED 0128 04F0 02C0 0904 0834	COMSPACE COMSTORE COMSVC COMSYSRV COMSYSTM COMSYSUT COMTABLE COMTIOT COMTOTAL COMTRACE	DS F DS F DS C DS CL8 EQU COMERG DS F DS T	BYTES AVAILABLE RESTORE CODE ADDRESS SVC FOR SORT SYSTEM RESERVED SYSTEM TYPE FROM CVT SYSOUT FILE NAME ES - 4 TRANSLATE/SIZE TABLE ADDRESS OF TIOT TOTAL RSA BYTES NOT IN OUTPUT RUN TRACE X'01'-X'0F' E15/E35 IN/OUT AREA RECOR LEVEL
082C 02A0 081C 02D2 02O4 0644 06E4 05B4	COMTRACK COMTRACT COMTRANS COMTREE COMTREE COMTREED COMTREED COMTREES	DS F DS F DS X DS H DS F DS 40F DS 16F DS 36F DS 20F	#TRACKS ALLOCATED FOR SORTWK EXTRACT ADDRESS TRANSFER INDICATION FOR ICEMESS CURRENT TRACK BALANCE 40F
0204 0208 0824 013A 08F0 0170 0158 091C	COMTREAV COMTSIZE COMTTIME COMTYPE COMUNBAL COMUNDO COMUTFIL COMVALCT	DS H DS F DS C DS F DS F DS F DS F	PREVIOUS TRACK BALANCE OFFSET TO LAST NODE CPU TIME WORK AREA FOR SMF TYPE MESSAGE CLASS ADDRESS OF UNBALANCED MERGE UNLOAD SUBROUTINE ADDRESS SORTOUT DCB ADDRESS SORTOUT BYTE COUNT BVAL STANDARD COUNT
01B8 0829 0003 00E4 05FC 00E6 0194 03AE	COMVALPR COMVBLKS COMVERSE COMVERZN COMVOC COMVOKED COMVOLCT COMWDXLS	DS F C S DS H C C C DS DS H	BVAL PRIOR COUNT N=BYPASS VLR BLOCKSET CURRENT COMMON VERSION NUMBER CONTAINS COMVERSE VIO/NOVIO SWITCH USER INVOKED STATUS SORTIN VOLUME COUNT WORKFILE DXLS
02E4 015C 0914 0234 08DC 08CC 02F6 02DF 02E2	COMWJOB COMWKDCB COMWKFIL COMWKLOC COMWKSIZ COMWMSSS COMWMNUM COMWMTAG	EQU COMMPF	SORTWK OPEN LIST ADDRESS BYTES IN WORK FILE SORTWK DATA ADDRESS IN IOX WORK BUF TO RSA ROUTINE SIZE OF WORK AREA EP + 9 MESSAGE
02DC 02B0 08C4 0354 02ED 02DC 02D8 03B8 0004	COMWPFIX COMWRECZ COMWRES COMWROUT COMWSTEP COMWTEXT COMWTEXT COMWTEXT COMWTEXT	EQU COMWTE DS F DS F EQU COMWJO CL120 DS F DS CL6 EQU 4	XT MESSAGE PREFIX SIZE OF A WORKFILE RECORD START OF RESIDUAL WORK AREA ROUTINE FOR WTO
0010 0001 0002 0371 02D6 0160 0424 0724	COMXEOS COMXESS COMXESS COMXSTAT COMXTENT COMOIDCB COM2NDRY COM43LST	EQU 16 EQU 1 EQU 2 EQU COMISC DS H DS F DS F DS OF	E03 IS PRESENT E15 IS PRESENT E35 IS PRESENT

## COMMA CROSS-REFERENCE TABLE (PEERAGE/VALE)

The following table contains the names of the COMMA fields together with their displacements.

			COMMENT  TO PREV SEC. ENTR OF ABENDR ADDR. OF TREEADD ROUTINE TREEGEN WORK CONSTANT ADDR. OF TREEADDS ROUTINE TREEGEN WORK AREA ALL WORKFILES ALLOCATED AVGERAGE RECORD LENGTH ADDR. OF ROUTINE BALLOCAT BASE-BIN MIN ALLOC BASE-BIN SIZE APTM411M ADDR TO BINBLK Q RSA BIN SIZE FOR MERGING MEMBER NAME FOR SORTIN INPUT BLOCKSIZE NEXT UNUSED ITTR SIZE OF FREE BLOCK LIST BLKS/TRK LOACALLY Y=BLOCKSET MAY BE USED ADDR. OF TREEGEN BR-TABLE ADDR. OF TREEGEN BR-TABLE ADDR. OF TREEGEN BR-TABLE ADDR. OF TREEGEN BR-TABLE ADDR. OF CURRENT WORK BUF IN - PTR IN BUFF TO VLR ICEBUG AREA ADDRESSES B37 RECOVERY MAG FLAG AL5A C=TRANSLATE CH AND AQ Y=CHECK COUNTERS YES = TAKE CHECKPOINTS INDEX AREA 1 HEAD INDEX AREA 2 HEAD INDEX AREA 2 HEAD INDEX AREA 2 HEAD CONCATENATION OF DATASETS Y=PEER N=NONPEER 0=5734 CONSTANT = -4 Y = MESSAGES TO CONSOLE CONSTANT = 256 ORIGINAL SIZE PARM Y = PRINT CRITICAL MSG ADDR. OF ROUTINE DALLOCAT DATA BLOCK SIZE ICEMAN BASEREGS ADDR. OF OPEN/CLOSE LIST PROC.FIRST DDNAME CHARACTER DEBUG SWITCH BYTE ADDR. OF SORTIN/OUT DECB USED IN ICELINK ICEDEC - ICEDED COMM REC LEN1 IN DEC REC LEN2 IN DEC REC LEN3 IN DEC REC LEN3 IN DEC REC LEN1 IN DEC REC LEN3 IN DEC REC LEN3 IN DEC REC LEN3 IN DEC REC LEN3 IN DEC REC LEN3 IN DEC REC LEN3 IN DEC REC LEN3 IN DEC REC LEN3 IN DEC REC LEN3 IN DEC REC LEN3 IN DEC REC LEN5 OR TREELETE ROUTINE PIR TO DEV WORK AREA ADDR. OF TREELETE ROUTINE PIR TO DEV WORK AREA ADDR. OF TREELETE ROUTINE PIR TO DEV WORK AREA ADDR. OF TREELETE ROUTINE PIR TO DEV WORK AREA ADDR. OF TREELETE ROUTINE PIR TO DEV WORK AREA ADDR. OF TREELETE ROUTINE PIR TO DEV WORK AREA ADDR. OF TREELETE ROUTINE PIR TO DEV WORK AREA ADDR. OF TREELETE ROUTINE PIR TO DEV WORK AREA ADDR. OF TREELETE ROUTINE PIR TO DEV WORK AREA ADDR. OF TREELETE ROUTINE PIR TO DEV WORK AREA ADDR. OF TREELETE ROUTINE PIR TO DEV WORK AREA ADDR. OF TREELETE ROUTINE PIR TO DEV WORK AREA ADDR. OF TREELETE ROUTINE PIR TO DEV WORK AREA
	LADEL 	ALIK	COUNTRY
5E5	CABRETRN	BIT(1)	TO PREV SEC. ENTR OF ABENDR
188	CADDISP	FIXED(31)	TREEGEN WORK CONSTANT
18C	CADDS	PTR(31)	ADDR. OF TREEADDS ROUTINE
3F4	CADDTAB	FIXED(31)	TREEGEN WORK AREA
495	CALLOCHD	BIT(1)	ALL WORKFILES ALLUCATED
154	CRALADDE	PIXED(31)	ANDE OF POULTINE BALLOCAT
190	CBASEMIN	FIXED(15)	BASE-BIN MIN ALLOC
3A4	CBASESIZ	FIXED(31)	BASE-BIN SIZE aPTM411M
194	CBINBLK	PTR(31)	ADDR TO BINBLK Q
198 750	CRIDINAM	CHVB(8)	MEMBER NAME FOR SORTIN
3E4	CBLKIN	FIXED(15)	INPUT BLOCKSIZE
19C	CBLKITTR	PTR(31)	NEXT UNUSED ITTR
144	CBLKLSIZ	FIXED(31)	SIZE OF FREE BLOCK LIST
5AU 344	CBLKPIRK	LIXED(31)	SEKSZIKK LUAGALLY Y=RIOCKSET MAY RE HSED
1 F4	CBRANCHP	PTR(31)	ADDR. OF TREEGEN BR-TABLE
1A0	CBUFFER	PTR(31)	ADDR. OF CURRENT WORK BUF
1 48	CBUFPTR	PTR(31)	IN - PTR IN BUFF TO VLR
/AC	CBUGADDR	FIXED(31)	RIT PECOVERY MAG FLAG 2154
347	CCHALT	CHAR(1)	C=TRANSLATE CH AND AQ
34A	CCHECK	CHAR(1)	Y=CHECK COUNTERS
4A0	CCHECKPT	CHAR(1)	YES = TAKE CHECKPOINTS
1AC	CCHIDX1	F1XED(31)	INDEX AREA I HEAD
1B4	CCHIDX3	FIXED(31)	INDEX AREA 2 HEAD
. 1B8	CCHIDX4	FIXED(31)	POOL AREA 1 HEAD
1BC	CCHIDX5	FIXED(31)	POOL AREA 2 HEAD
495 443	CCUNCASW	BII(1)	CUNCATENATION OF DATASETS
104	CCONNEG4	FIXED(31)	CONSTANT = -4
330	CCONSMOD	CHAR(1)	Y = MESSAGES TO CONSOLE
1C0	CCON256	FIXED(31)	CONSTANT = 256
108	CCURESAV	FIXED(31)	ORIGINAL SIZE PARM
1D4	CCURRRIK	PTR(31)	ADDR TO CURR RINRIK
150	CDALADDR	PTR(31)	ADDR. OF ROUTINE DALLOCAT
1E4	CDATLEN	FIXED(31)	DATA BLOCK SIZE
100	CDAILUC	BDY(WORD)	ICEMAN BASEREGS .
494	CDDFIRST	BIT(1)	PROC FIRST DONAME CHAPACTED
21E	CDEBUG	CHAR(1)	DEBUG SWITCH BYTE
1E8	CDECAREA	PTR(31)	ADDR. OF SORTIN/OUT DECB
/84 588	CDECAREB	PTR(31)	USED IN ICELINK
57C	CDECL1	FIXED(31)	REC   EN1 IN DEC
580	CDECL2	FIXED(31)	REC LEN2 IN DEC
584	CDECL3	FIXED(31)	REC LENS IN DEC
1F0	CDELCAT	L1YED(21)	SIZE OF SURIIN/OUT DECB
1F4	CDELETE	PTR(31)	ADDR. OF TREELETE ROUTINE
588	CDEVWK	PTR(31)	PTR TO DEV WORK AREA
1F8	CDEXCTR	FIXED(31)	HI/LO INDEX COUNTER
110	CDEYTLIK	PIK(31)	THIRK BLUCK TILK

DISP	LABEL	ATTR	COMMENT
200 204	CDEXPBUF	FIXED(31)	INDEX ENTRIES PER BUFFER INDEX ENTRIES PER TRACK YES = PRINT DIAGNOSTICS HI CONTROL FIELD FOR IMS DISP IS MOD FOR SORTOUT SORTDK ALLOCATED
49E	CDIAGMOD	CHAB(1)	THUEX ENIKIES PEK IKACK YES = PRINT DIAGNOSTICS
218	CDIMSHI	FIXED(16)	HI CONTROL FIELD FOR IMS
7E2	CDISPMOD	CHAR(1)	DISP IS MOD FOR SORTOUT
496	CDKALLOC	BIT(1)	SORTDK ALLOCATED
6EC 208	CDYNAMIC	CHAR(1)	
496	CDYNAMIC CDYNGEN CDYNLEN CDYNUMB CDYNUMB	BIT(1)	ADDR OF UNUSED DYNAMIC AREA DYNALLOC GENERIC NAMES USED DYNALLOC FIELD LENGTH NUMB OF DYN ALLOC DEVICES
218	CDYNLEN	PTR(8)	DYNALLOC FIELD LENGTH
214	CDYNNUMB	FIXED(31)	NUMB OF DYN ALLOC DEVICES
495 20C	CDANMK	RTI(1)	UNE DYNALLIE: PALLED
20C	CDYNWK1	CHAR(8) PTR(8)	1ST BYTE OF DYN DEV TYPE
21C	CEBUFFRS	FIXED(16)	NUMBER OF E-PHASE BUFFERS
228	CEMERGE	BIT(1) CHAR(8) PTR(8) FIXED(16) FIXED(31) CHAR(1) PTR(31) PTR(31)	MERGE ORDER - ELIMINATION
21F 178	CENDITUDE	CHAR(1) PTD(31)	NO = UNEXPECTED SYSIN EOF WORK AREA FOR EXTRACT
794	CEQE25	PTR(31)	PICK UP EQUALS FIELD
794	CEQE250P	CHAR(2)	OP CODE ICM
334	CEQUALS	CHAR(1)	Y = PRESERV ORDER OF EQUALS
337 494	CERKINA	CHAR(1)	Y = CRITICAL ERROR ABEND
494	CERRSW2	BIT(1)	ERROR IN REQUIRED PARAMETER ERROR IN OPTIONAL PARAMETER
336	CERRU016	CHAR(1)	V ~ COTTICAL EDDAD ABEUR
4A8	CESTIMSZ	CHAR(1)	YES = ESTIM. SIZE FILSZ
4A7 7AA	CEXACISZ	CHAR(1) CHAR(1) CHAR(1)	YES = EXACT FILSZ GIVEN EXCP CAN BE USED FOR SORTIN
7 A B	CEXCPOUT	CHAR(1)	EXCP -//- SORTOUT
790	CEXITDEL	PTR(31)	EXIT DELETE ROUTINE ADDR.
78C	CEXITLDA	PTR(31)	EXIT DELETE ROUTINE ADDR. EXIT LOAD ROUTINE ADDRESS LNG OF PH1 RTNS
22C 230	CEXTIPHI	FIXED(31)	LNG OF PH1 RTNS LNG OF PH2 RTNS
234	CEXITPH3	FIXED(31)	LNG OF PH3 RTNS
57C	CEXITHS CEXITIAB CEXITIS CEXITIS CEXITIS CEXITIS CEXITIS CEXITIS CEXITIS CEXITIS CEXITIS CEXITIS CEXITIS CEXITIS CEXITIS CEXITIS CEXITIS	CHAR(1) PTR(31) PTR(31) CHAR(2) CHAR(1) BIT(1) BIT(1) CHAR(1) CHAR(1) CHAR(1) CHAR(1) CHAR(1) FIXED(31) FIXED(31) FIXED(31) FIXED(31) CHAR(16)	EXIT RETURN BRANCHTAB
23C	CEXIT11	BDY(WORD)	
248 254	CEXTILE	BDY(WORD) BDY(WORD)	E15/E32 EXIT - E16 EXIT
260	CEXIT17	BDY(WORD)	E17 EXIT
26C	CEXIT18	BDY(WORD)	E18 EXIT
284	CEXIT19	BDY(WORD)	E19 EXIT
29C 768	CEXTI21	BDY(WORD) BDY(WORD)	E21 EXIT EXIT25
2A8	CEXIT27	BDY(WORD)	E27 EXIT
284	CEXIT21 CEXIT25 CEXIT27 CEXIT31 CEXIT35 CEXIT37 CEXIT38 CEXIT39 CEXIT61	BDY(WORD)	E31 EXIT
200	CEXIT35	BDY(WORD)	E35 EXIT -
2CC 2D8	CEXT13/	BDY(WORD) BDY(WORD)	CE37 EXIT E38 EXIT
2E4	CEXIT39	BDY(WORD)	E39 EXIT
290	CEXIT61	BDY(WORD)	E61 EXIT
341	CEXMODE	CHAR(1)	R=EXCPVR V=EXCP X=MVT
224 220	CEXPAND CEXTRACT	FIXED(31) PTR(31)	INT LRECL EXPAND POS OR NEG ADDR: OF EXTRACT ROUTINE
244	CETICALL	PTR(31)	CALL ADDRES
244	CE11LNG	FIXED(31)	EXIT LENGTH
23C	CELEADDB	CHAR(8)	NAME BASSED ADDRESS
248 250	CE15ADDR CE15CALL	PTR(31) PTR(31)	PASSED ADDRESS CALL ADDRESS
6 D 4	CEISDELT	BIT(1)	1=E15 DELETED
250	CE15LNG	FIXED(31)	LENGTH
4A4 248	CE15MOD	CHAR(1)	YES = E15 ADDRESS GIVEN NAME
248 58C	CE15NAME CE15PARM	CHAR(8) CHAR(4)	E15 PARAMEYER LIST
25C	CE16CALL	PTR(31)	CALL ADDR
25C	CE16LNG	FIXED(31)	E16 LENGTH

DISP	LABEL	ATTR	NAME CALL ADDRESS E17 LENGTH NAME CALL ADDRESS USER EOD ADDR ADDR TO VSAM EXIT LIST E18 LENGTH NAME ADDR TO VSAM PASSW. LIST ADDR TO USER XLIST(5) RTN CALL ADDRESS E19 LENGTH NAME CALL ADDR E21 LENGTH NAME CALL ADDR E21 LENGTH NAME CALL ADDR E27 LENGTH EXIT E25 LENGTH EXIT E25 NAME CALL ADDR E27 LENGTH NAME CALL ADDRESS E31 LENGTH NAME CALL ADDRESS E31 LENGTH NAME E32 MERGE ORDER PASSED ADDRESS
254	CE16NAME	CHAR(8)	NAME
268 268	CE17CALL	PIK(31)	E17 LENGTH
260	CE17LNG CE17NAME	CHAR(8)	NAME
274	CE18CALL	PTR(31)	CALL ADDRESS
238	CE18EOD	PTR(31)	USER EDD ADDR
27C 274	CE18EXL CE18LNG	FIXED(31)	E18 LENGTH
26C	CE18NAME	CHAR(8)	NAME
280	CE18PSW_	PTR(31)	ADDR TO VSAM PASSW. LIST
278 28C	CE18XLST	PTR(31)	CALL ADDRESS
28C	CE19LNG	FIXED(31)	E19 LENGTH
284	CE19NAME	CHAR(8)	NAME
2A4	CE21CALL	PTR(31)	CALL ADDR
2A4 29C	CEZILNG	LIXED(21)	NAME NAME
770	CE25CALL	PTR(31)	AND ITS CALL ADDRESS
770	CE25LNG	FIXED(31)	EXIT E25 LENGTH
768	CE25NAME	CHAR(8)	EXIT E25 NAME
2B0 2B0	CE2/CALL CE27ING	FIXED(31)	F27 I FNGTH
288	CE27NAME	CHAR(8)	NAME
2BC	CE31CALL	PTR(31)	CALL ADDRESS
2BC	CE31LNG	FIXED(31)	E31 LENGTH
2B4 24C	CESTNAME CESTNAME	FTYFD(31)	F32 MERGE ORDER
24D	CE32M024	PTR(24)	LOC HERVE WINDER
200	CE35ADDR	PTR(31)	PASSED ADDRESS
5F4 2C8	CE35AREA	PTR(31)	CALL ADDDESS
208	CESSUALL CESSING	FIXED(31)	I FNGTH
445	CE35MOD	CHAR(1)	YES = E35 ADDRESS GIVEN
200	CE35NAME	CHAR(8)	E27 LENGTH NAME CALL ADDRESS E31 LENGTH NAME E32 MERGE ORDER  PASSED ADDRESS  CALL ADDRESS LENGTH YES = E35 ADDRESS GIVEN NAME E35 PARAMETER LIST CALL ADDRESS E37 LENGTH NAME CALL ADDRESS E38 LENGTH NAME CALL ADDRESS E38 LENGTH NAME CALL ADDRESS E38 LENGTH NAME CALL ADDRESS E39 LENGTH NAME CALL ADDRESS E39 LENGTH NAME
590 2D4	CESSPARM CESTCALL	PTR(31)	CALL ADDRESS
2D4	CE37LNG	FIXED(31)	E37 LENGTH
200	CE37NAME	CHAR(8)	NAME
2E0 2E0	CE38CALL	PTR(31)	CALL ADDRESS
2D8	CE30LNG CE38NAME	CHVER(8)	NAME
2EC	CE39CALL	PTR(31)	CALL ADDRESS
2F0	CE39EXL	PTR(31)	ADDR TO VSAM EXIT LIST
2EC 2E4	CESPLNG	FIXED(31)	E39 LENGTH
2F4	CE39PSW	PTR(31)	ADDR TO VSAM PASSW. LIST ADDR TO USER XLIST(5) RTN CALL ADDR E61 LENGTH NAME REC ADDRESS
2F8	CE39XLST	PTR(31)	ADDR TO USER XLIST(5) RTN
298	CE61CALL	PTR(31)	CALL ADDR
298 290	CEGILNG CEGINAME	CHAB(8)	NAME FOT FEUGLA
270	CE61PADD	FIXED(31)	REC ADDRESS
2FC	CE61PARM		E61 PARM LIST
2FF	CE61PCF CE61PLEN	CHAR(1) FIXED(31)	CONTR FIELD NUMB
496	CFAKE35	BIT(1)	CONTR FIELD LEN L2 GET L3 SWITCH
308	CFIELDS	FIXED(31)	NUMBER OF CONTROL FIELDS
804 310	CFILBYTE	FIXED(31)	FILE SIZE ACCUM (BYTES)
310 30C	CFILIMIT CFILSIZE	PTR(31) FIXED(31)	ADDR. OF END OF BUFFER USER SPECIFIED FILE SIZE
30D	CFILSZ24	PTR(24)	THREE BYTES
495	CFIRSTSW	BIT(1)	OPEN OF FIRST DATA SET
495	CFIRSTSB CFIXPAGE	BIT(1) BIT(1)	1=FIRST SUB-BIN PROTECTED AREA FIXED
5D7	CFIXREC	BIT(1)	ON = FIXED LENGTH RECORDS
314	CFREECTR	FIXED(31)	AVAILABLE RSA BIN COUNT

DISP	LABEL	ATTR	COMMENT  ADDR. OF FIRST FREE AREA START OF FREE SPACE CHAIN ADDR. OF ROUTINE FREESTOR SIZE OF ZEROTH FREE AREA 1=FIRST BIN ALLOC ADDR. OF ROUTINE GALLOCAT SORT GENERATED DEFAULTS: SORT GEN DEFAULTS: SORT GEN DEFAULTS CONT: ADDR. OF ROUTINE GETMAINS ADDR. OF TEMP. GETM. AREAS PERM AREAS CHAIN ADDR. OF INPUT GET ROUTINE ALTERNATE NAME INDEX IN GETMAIN TABLE INDEX HOR BFR ADDR CHECKPOINT DCB POINTER TOTAL BLKS EST FOR PRIM EXT CHKPT ITT SAVE AREA INPUT BFR SPACE ADDR. OF INPUT INDEX BUFFER INDEX ENTRY SIZE INDEX BLOCK SIZE BLOCKS/LOGIC INDEX SORT INVOKED BY IMS NO CHAIN SCHED FOR SORTIN CONCATENATED SORTIN FLAG 1=INCORE SORT ADDR. OF INPUT INDEX BUF BUFFER POINTERS ADDR TO INPUT REC SORTIN LOGICAL RCD LENGTH TREEGEN WORK AREA YES = DYNAMIC INVOCATION HOLD SW/Q END ADDR OF INIT ROUT ADDR. OF ROUTINE INOUTRTN I/O TASK SAVE AREA YES = DYNAMIC INVOCATION HOLD SW/Q END ADDR OF FREE TRACK HOLDERS ITTR PIT TO 1ST INP STRING ALTERNATE ITTR PTR HEAD OF AVAIL. TRACK QUEUE JOB NAME NUMBER OF INDEXES PER TRACK SIZE OF CONTROL WORD LAST ENTRY TO REAL TIME RTN # OF LA-INSTR. IN RCD-MOVE LENGTH LAST CLC. TREEGEN SMF WORK AREA PIR ZERO IF NO MORE MOD TO CALL FLAGS LINKEDITING REQUEST
180	CFREHEAD	PTR(31)	ADDR. OF FIRST FREE AREA
166	CERECTOR	BUT(UWUKU)	STAKE OF POULTINE EDUCATION
184	CERESIUR	FIXED(31)	ADDR. OF KUUIINE FREESIUK
6 D 4	CFRSTRIN	RTT(1)	1=FIRST RIN ALLOC
148	CGALADDR	PTR(31)	ADDR. OF ROUTINE GALLOCAT
320	CGENDOPT	BDY(WORD)	SORT GENERATED DEFAULTS:
32C	CGENDOP1	BDY(WORD)	SORT GEN DEFAULTS CONT:
140	CGETMAIN	PTR(31)	ADDR. OF ROUTINE GETMAINS
354	CGETMTMP	PTR(31)	ADDR. OF TEMP. GETM. AREAS
356	CGETPEO	PIR(31)	PERM AREAS CHAIN
356	COETEMP	PIR(31)	AUDK. OF INPUT GET RUUTINE
318	CGMINDEX	FIXED(31)	TNDEY IN GETMAIN TARIE
368	CHBUF1	FIXED(31)	INDEX HDR BFR ADDR
360	CHECKDCB	FIXED(31)	CHECKPOINT DCB POINTER
36C	CHKBLK	FIXED(31)	TOTAL BLKS EST FOR PRIM EXT
364	CHKTTSTR	PTR(31)	CHKPT ITT SAVE AREA
378	CIBSPACE	FIXED(31)	INPUT BFR SPACE
3/4 386	CIBULLFK	PIR(31)	ADDR. OF INPUT INDEX BUFFER
307	CIDEX215	LIXED(31)	INDEX ENIKY SIZE
352	CIDXPLB	FIXED(31)	BIUCKS/IUCIC INDEA
495	CIMS	BIT(1)	SORT INVOKED BY TMS
496	CINCHAIN	BIT(1)	NO CHAIN SCHED FOR SORTIN
7F8	CINCONCA	CHAR(1)	CONCATENATED SORTIN FLAG
6D4	CINCORE	BIT(1)	1=INCORE SORT
3/0	CINDEX	PTR(31)	ADDR. OF INPUT INDEX ENTRY
78 N	CINDEANI	PIK(31)	ADDR. OF END OF INDEX BUF
77C	CINDEXII	PTR(31)	RUFFED POINTERS
394	CINLOC	FIXED(31)	ADDR TO INPUT REC
370	CINSIZE	FIXED(31)	SORTIN LOGICAL RCD LENGTH
18C	CINSTRLT	FIXED(31)	TREEGEN WORK AREA
4 7 F	CINVOKED	CHAR(1)	YES = DYNAMIC INVOCATION
100	CIONULD	EIAED(31)	NOTE OF THIS BOILS
15C	CIORADDR	PTR(31)	ADDR OF BUILTINE INCUIDED
1D0	CIOSAV13	FIXED(31)	I/O TASK SAVE AREA
388	CITTFQ	PTR(31)	HEAD OF FREE TRACK HOLDERS
214	CITTR	FIXED(31)	ITTR PTR TO 1ST INP STRING
210	CITTRA	FIXED(31)	ALTERNATE ITTR PTR
38C	CTLIMG	PTR(31)	HEAD OF AVAIL. TRACK QUEUE
4 A A	CACAMUDE	PTP(R)	JUB NAME NUMBED DE INDEVES DED IDAAV
398	CKEYSIZE	FIXED(31)	SIZE OF CONTROL WORD
3A8	CLASTIME	FIXED(31)	LAST ENTRY TO REAL TIME RTN
1EC	CLATCHT	FIXED(31)	# OF LA-INSTR. IN RCD-MOVE
3E8	CLCLTH	FIXED(31)	LENGTH LAST CLC. TREEGEN
3E4	CLCNUM	FIXED(31)	NO. OF CLC-INSTR. TREEGEN
7FC 41C	CLENSTAT CLINK	PTR(31) FIXED(31)	SMF WORK AREA PTR ZERO IF NO MORE MOD TO CALL
778	CLINKREQ	CHAR(1)	FLAGS LINKEDITING REQUEST
7A8	CLISTMAD	CHAR(1)	TEMP SAVE FOR CLISTMOD
335	CLISTMOD	CHAR(1)	Y = LIST CONTROL CARDS
3AC	CLOCKS		TIME ACCUMULATORS
3AC	CLOCKO	FIXED(31)	C-PHASE ELAPSED TIME
3B0	CLOCK1	FIXED(31)	C-PHASE CPU TIME
3D4 3D8	CLOCK10 CLOCK11	FIXED(31)	SPECIAL CLOCK 2
3DC	CLOCK11 CLOCK12	FIXED(31) FIXED(31)	SPECIAL CLOCK 3 SPECIAL CLOCK 4
3E0	CLOCK12	FIXED(31)	SPECIAL CLOCK 5
3B4	CLOCK2	FIXED(31)	P-PHASE ELAPSED TIME
3B8	CLOCK3	FIXED(31)	P-PHASE CPU TIME
3BC	CLOCK4	FIXED(31)	R-PHASE ELAPSED TIME

DISP LABEL ATTR COMMENT  3C0 CLOCK5 FIXED(31) R-PHASE CPU TIME 3C4 CLOCK6 FIXED(31) E-PHASE ELAPSED TIME 3C8 CLOCK7 FIXED(31) E-PHASE CPU TIME 3C8 CLOCK8 FIXED(31) SPECIAL CLOCK 0 3D0 CLOCK9 FIXED(31) SPECIAL CLOCK 1 39C CLOCSIZE BDY(WORD) TABLE FOR VARIABLE GETM/ 779 CLODCNT1 CHAR(1) LOAD COUNT PHASE 1 774 CLODCNT2 CHAR(1) LOAD COUNT PHASE 2 775 CLODCNT3 CHAR(1) LOAD COUNT PHASE 3 3C6 CLTHIN FIXED(15) INPUT RECORD LENGTH 408 CMAINSIZ FIXED(31) SIZE OF WORKING STORAGE 31C CMAMMSG FIXED(31) PIR TO MSGTABLE FOR REAL 60C CMAXREC FIXED(31) MAXIMUM RECORD SIZE 404 CMAXSIZE FIXED(31) MAXIMUM RECORD SIZE 404 CMAXSIZE FIXED(31) MAXIMUM SIZE WANTED 35C CMERGE PIR(31) ADDR. OF TEMBERGE ROUT: 5E5 CMERGEIN BIT(1) SORTINO1 DO STMNT PRESEN 400 CMINAX BDY(WORD) PARMS FOR VARIABLE GETM/ 400 CMINREC FIXED(31) MINIMAL RECORD SIZE 400 CMINREC FIXED(31) MINIMAL RECORD SIZE 400 CMINREC FIXED(31) MINIMAL RECORD SIZE 400 CMINREC FIXED(31) MINIMAL RECORD SIZE 400 CMINSIZE FIXED(31) MINIMAL RECORD SIZE 400 CMINSIZE FIXED(31) MINIMAL RECORD SIZE 400 CMINSIZE FIXED(31) MINIMAL RECORD SIZE 400 CMINSIZE FIXED(31) MINIMAL RECORD SIZE 400 CMINSIZE FIXED(31) MINIMAL RECORD SIZE 400 CMODDCB FIXED(31) MODSLIB DCB ADDRESS 410 CMODDCB FIXED(31) MODSLIB DCB ADDRESS 411 CMODDAME CHAR(3) SIGNIFICANT PART OF MODI 412 CMODDCB FIXED(31) MODSLIB PART NAME 414 CMODSTRT PIR(31) STARTING BYTE 415 CMODDAME CHAR(3) SIGNIFICANT PART OF MODI 416 CMODDER FIXED(15) HALF—MORD 417 CMODULE CHAR(8) MOD TO BE CALLED 418 CMODSEND PIR(31) FIXED(15) HALF—MORD 419 CMOVERN PIR(31) FIXED(15) HALF—MORD 410 CMOUVERR (3) PIR(16) CTL WORD MYC CREAT TO BUFFER TO RSA 416 CMOVERB (3) PIR(16) CTL WORD MYC OR BAL TO FIXED(15) CTL WORD MYC OR BAL TO FIXED(15) CTL WORD MYC OR BAL TO FIXED(15) COUNT OF ACTION MESSAGE BLANK 414 CMOVERB (3) PIR(16) CTL WORD MYC OR BAL TO FIXED(15) CMOVE OR BAL TO FIXED(15) CMOVE OR BAL TO FIXED(15) CMOVE OR BAL TO FIXED(15) CMOVE OR BAL TO FIXED(15) CMOVE OR BAL TO FIXED(15) CMOVE OR BAL TO FIXED(15) CMOVE OR BAL TO FIXED(15) CMOVE OR BAL TO FIXED(15) C	
3C0 CLOCK5 FIXED(31) R-PHASE CPU TIME 3C4 CLOCK6 FIXED(31) E-PHASE ELAPSED TIME	
3C0 CLOCKS FIXED(31) R-PHASE CPO TIME 3C4 CLOCK6 FIXED(31) E-PHASE ELAPSED TIME	
3C4 CLUCK6 FIXED(31) E-PHASE ELAPSED TIME	
300 CLOCK) LIVED(31) SECURIC CLO IINE	
200 CLOCKO EIAED(21) SEECIME COOK A	
TARLE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE STORY OF THE	TN
779 CLODENT1 CHAR(1) LOAD COUNT PHASE 1	1211
774 CLODONT2 CHAR(1) LOAD COUNT PHASE 2	
77B CLODENTS CHAR(1) LOAD COUNT PHASE 3	
3E6 CLTHIN FIXED(15) INPUT RECORD LENGTH	
408 CMAINSIZ FIXED(31) SIZE OF WORKING STORAGE	
31C CMANMSG FIXED(31) PTR TO MSGTABLE FOR RES	<b>TORE</b>
324 CMAXLIM FIXED(31) MAX LIMIT FOR SIZE(MAX)	
60C CMAXREC FIXED(31) MAXIMUM RECORD SIZE	
404 CMAXSIZE FIXED(31) MAXIMUM SIZE WANTED	
3F4 CMERGE PTR(31) ADDR. OF TRMERGE ROUT	INE
5E5 CMERGEIN BIT(1) SURTINGI DD STMNT PRESEI	NT.
6DC CMESSAGE CHAR(120) MESSAGE AREA	
328 CMINLIM FIXED(31) MIN VIRTUAL SIRG FOR PER	EK .
400 CMINMAX BDI(WARD) PARMS FUR VARIABLE GEIM	ATM
ADE CHINKEC FIXED(31) MINIMAL RECORD SIZE	
ADD CHINCED FIXED(13) HARF-WARD	
446 CMODCARD CHAR(1) YES = MODS STMNT PRESENT	r
7BO CMODDCB FIXED(31) MODSLIB DCB ADDRESS	•
410 CMODEVAL FIXED(31) MODAL RECORD SIZE	
41F CMODNAME CHAR(3) SIGNIFICANT PART OF MODI	MAME
418 CMODSEND PTR(31) ENDING BYTE	
448 CMODSLIB CHAR(8) MODS LIBRARY NAME	
414 CMODSTM BDY(WORD) MODS STATEMENT -	
414 CMODSTRT PTR(31) STARTING BYTE	
41C CMODULE CHAR(8) MOD TO BE CALLED	
412 CMODVALH FIXED(15) HALF-WORD	
SPC CMUREIRK PIK(SI) IKKS NUI 10 KELEASE	
430 CMOVEDRICAL PIRCID) CCD MVC BUFFER IU RSA	
178 CMOVEDRICS) FIRCID) INDEX NVC DUFFER ID RSA	
442 CMOVEERS TRUST MORE AREA TOR MOVECOME	
17C CMOVEH PTR(31) WORK AREA FOR MOVECOMP	
3EC CMOVEKEY(3) PTR(16) CTI WORD MVC OR BAI TO I	?TN
454 CMOVELNG FIXED(31) BYTES TO MOVE TO RSA	.,,,
424 CMOVERB (3) PTR(16) RCD MVC RSA TO BUFFER	
42A CMOVERBC(3) PTR(16) RCD MVC RSA TO BUFFER(C-	-PH)
43C CMOVERBI(3) PTR(16) INDEX MVC RSA TO BUFFER	
7FA CMSGACNT FIXED(15) COUNT OF ACTION MESSAGES	3
160 CMSGADDR PTR(31) ADDR. OF ROUTINE DIAGNO	DSE
OFF CMSGBLNK CHAR(1) MESSAGE BLANK	
OFD CMSCLICT (20) ELVER(34) MESSAGE LEXT	
4DE CMSCNO CUADCA MEGGACIANTE INTU LISI	
330 CMSGOPIN CHAP(4) MESSAGE OPTION	
450 CMSGTABL PTR(31) ADDR. ERROR MESSAGE TABI	_
6E2 CMSGTAG CHAR(1) MESSAGE TAG	. C.
3F8 CMSGTYPE PTR(31) DEBUG MESSAGE TYPE INDI	ŦX
763 CMULIVOL CHAR(1) MULTIVOLUME DATA SET FL/	ĬĜ
494 CNEWPAGE BIT(1) NEW PAGE HAS BEEN TAKEN	. =
45C CNEXTWRK FIXED(31) PRIM WORKD CHAIN	
496 CNOREUSE BIT(1) VSAM OUTPUT NON-REUSABLE	<b>:</b>
458 CNUMWORK FIXED(31) NUMBER OF WORK AREAS	
TED AGRARIAN APRILATE STATES	DE
7E8 COBSBLOK PTR(31) ADDRESS OF EXCP DATA CO	YTT
7E8 COBSBLOK PTR(31) ADDRESS OF EXCP DATA CO 7D8 COBSEXIN PTR(31) ADDRESS OF SORTIN OPEN E	-/
7E8 COBSBLOK PTR(31) ADDRESS OF EXCP DATA CO 7D8 COBSEXIN PTR(31) ADDRESS OF SORTIN OPEN E 7E4 COBSEXIT PTR(31) EXCP OPEN EXIT ADDRESS	
7E8 COBSBLOK PTR(31) ADDRESS OF EXCP DATA CO 7D8 COBSEXIN PTR(31) ADDRESS OF SORTIN OPEN E 7E4 COBSEXIT PTR(31) EXCP OPEN EXIT ADDRESS 7DC COBSEXUT PTR(31) ADDRESS OF SORTOUT -//	
7E8 COBSBLOK PTR(31) ADDRESS OF EXCP DATA CO 7D8 COBSEXIN PTR(31) ADDRESS OF SORTIN OPEN E 7E4 COBSEXIT PTR(31) EXCP OPEN EXIT ADDRESS	<b>'-</b>

DISP	LABEL		COMMENT
39C	COMLOC	PTP(31)	GETMAINED CHAIN COREPARM SAVE FOR ICEDEV EXTRACT/RESTORE CODE AREA END OF EXTRACT/RESTORE AREA DOUBLEWORD-ALIGNED WORKAREA SIZE OF RETURNED STORAGE BLKSIZE OF 1-ST SORTIN FILE NO CHAIN SCHED FOR SORTOUT  MISCELLANEOUS SWITCHES MISCELLANEOUS SWITCHES MISCELLANEOUS SWITCHES MISCELLANEOUS SWITCHES OP/CL SYSOUT IN PRSS PERFORM OPEN IN ICEOBS OPEN PTR USED IN DEF + CRE OPTION SWITCHES MAXIMUM ALLOWED FOR SORT/MERGE CORE USED FOR MSG 92/93 CORE VALUE FOR MSG 39 (DEV)
174	COMLOCSV	FIXED(31)	CORFPARM SAVE FOR ICENEY
4B0	COMMB	PTR(31)	EXTRACT/RESTORE CODE AREA
4B4	COMMBEND	PTR(31)	END OF EXTRACT/RESTORE AREA
178	COMMDBLE	BDY(DWORD)	DOUBLEWORD-ALIGNED WORKAREA
3A0	COMSIZE	FIXED(31)	SIZE OF RETURNED STORAGE
38C	CUNBLKSZ	FIXED(15)	BLKSIZE OF 1-ST SORTIN FILE
496 4B8	CONCHAIN	BII(I)	NO CHAIN SCHED FOR SORTOUT
497	CONFRECEM	CHVEN(12)	
494	CONSW	PTR(8)	MISCELLANEOUS SUTTONES
495	CONSW1	PTR(8)	MISCELLANEOUS SWITCHES
496	CONSW2	PTR(8)	MISCELLANEOUS SWITCHES
5E5	CONSW3	PTR(8)	MISCELLANEOUS SWITCHES
496 7F9	COPENCLS	BIT(1)	OP/CL SYSOUT IN PRSS
464	COPERTIN	CHARLIJ	PERFURM UPEN IN ICEOBS
498	COPTIONS	LIK(21)	OPEN PTR USED IN DEF + CRE
320	COREPARM	FIXED(31)	MAXIMUM ALLOWED
321	COREPM24	PTR(24)	FOR SORT/MERGE
5B0	COREUSED	FIXED(31)	CORE USED FOR MSG 92/93
5A0	CORMSG39	PTR(31)	CORE VALUE FOR MSG 39 (DEV)
470 470	COUNTERS	ETVER/743	EVENT COUNTERS
474	COUNTERD	LIXED(31)	FULKIED EACHS
478	COUNTER2	FIXED(31)	RECORDS IN
47C	COUNTER3	FIXED(31)	RECORDS OUT
480	COUNTER4	FIXED(31)	RECORDS INSERTED
484	COUNTER5	FIXED(31)	RECORDS DELETED
488 48C	COUNTERS	FIXED(31)	SPECIAL COUNT O
490	COUNTERY	LIXED(31)	TEMPORARY COUNT 1
7Á4	COUNTERY	FIXED(31)	CORE USED FOR MSG 92/93 CORE VALUE FOR MSG 39 (DEV) EVENT COUNTERS ENTRIES TO INOUTRIN ISSUED EXCPS RECORDS IN RECORDS OUT RECORDS DELETED SPECIAL COUNT 0 SPECIAL COUNT 1 TEMPORARY COUNT 1 TEMPORARY COUNT 1 TEMPORARY COUNT 1 TEMPORARY COUNT 1 TEMPORARY COUNT 1 TEMPORARY COUNT 1 TEMPORARY COUNT 1 TEMPORARY COUNT 1 TEMPORARY COUNT 1 TEMPORARY COUNT 1 TEMPORARY COUNT 1 TEMPORARY COUNT 1 TEMPORARY COUNT 1 TEMPORARY COUNT 1 TEMPORARY COUNT 1 TEMPORARY COUNT 1 TEMPORARY COUNT 1 TEMPORARY COUNT 1 TEMPORARY COUNT 1 TEMPORARY COUNT 1 TEMPORARY COUNT 1 TEMPORARY COUNT 1 TEMPORARY TO UNPROT FIXED AREA ADDR. OF FIXED AREAS TABLE PTR TO END OF LAST BUFFDO PAGE SIZE FOR SYSTEM PAR INDEX BIN COMP PARTITION MODE SWITCH PAR INDEX BIN MOVE RESERVED
46C	COUTLIM	PTR(31)	ADDR. OF END OF WORK BUF
468	COUTSIZE	FIXED(31)	SORTOUT LOGICAL RCD LENGTH
4D0 4D4	CPAGELIM	PIR(31)	END PTR TO UNPROT FIXD AREA
5A0	CPAGERSA	PTP(31)	ADDR. OF FIXED AREAS TABLE PTR TO END OF LAST BUFFDO
4AC	CPAGESIZ	PTR(31)	PAGE SIZE FOR SYSTEM
4DC	CPARCLC (3)	PTR(16)	PAR INDEX BIN COMP
495	CPARMODE	BIT(1)	PARTITION MODE SWITCH
4E2	CPARMVC (3)	PTR(16)	PAR INDEX BIN MOVE
5A4	CPATCH CPBUF1 CPEERAGE CPHASECG CPHASEEG CPHASERG CPHASEWK	LIXED(31)	RESERVED INDEX POOL ADDR
4 D 8	CPEERAGE	FIXED(31)	MIN. NO STRINGS ON SORTWK
56C	CPHASECG	FIXED(31)	MAXIMUM G-VALUE - CREATION
578	CPHASEEG	FIXED(31)	MAXIMUM G-VALUE-ELIMINATION
56C 578 570 574	CPHASEPG	FIXED(31)	MAXIMUM G-VALUE - PARTTION
574 57C	CPHASEKG.	LIXED(21)	MAXIMUM G-VALUE - REDUCTION PHASE WORK AREA
7A9	CPHASLNK	CHAR(1)	TO REMEMBER THAT WE LINK
56B	CPREGION	CHAR(1)	REGION IS CORE LIMIT
332	CPRINMOD	CHAR(1)	Y = PRINT ALL MESSAGES
5AC	CPRMSIZ	FIXED(31)	PERM ALLOC SIZE
5A8	CPROTSIZ	FIXED(31) CHAR(129)	PROT AREA SIZE
4EA 4BC	CPRT CPRTDECB(5)	PTR(31)	(WORD 3) PRINT BUFFER PRINTER OUTPUT DECB
4EA	CPRTHEAD (4)	FIXED(15)	VLR AND VLB INFORMATION
4F2	CPRTLINE	CHAR(121)	(WORD 3) PRINTER OUTPUT
331	CPRTRMOD	CHAR(1)	Y = MESSAGES TO PRINTER
59C 4BA	CPUTREC CPOSIZE	PTR(31) FIXED(15)	ADDR. OF OUTPUT PUT ROUTINE PARTITION ZERO SIZE
14C	CRALADDR	PTR(31)	ADDR. OF ROUTINE RALLOCAT
5D4	CRBUFFRS	FIXED(16)	NUMBER OF R-PHASE BUFFERS
5B0	CREBUILD	PTR(31)	REBUILD SPANNED RECORDS

DISP	LABEL	ATTR	
5B8	CRECADDR	PTR(31)	ADDR. OF RECORD - FOR VSAM
5BC	CRECAREA	PTR(31)	VLR PAD/RESTORE AREA
500 350	CRECENTA	LIXED(21)	THRUT PECOPO FORMAT
168	CRECHAIN	PTR(31)	ADDR. OF ROUTINE RECHAIN
5C4	CRECORD	FIXED(31)	ADDR TO RECORD KEY
5CC	CRECPBUF	FIXED(31)	RECORDS/BUFFER
5C8	CRECPTR	PTR(31)	SPANNED REC PTR
584	CRECSIZE	FIXED(31)	SIZE OF TRANSPOSED RECORD
507	CRECIBUR	bib(8) Liyen(31)	RECORD TYPE
343	CRELSE	CHAR(1)	Y=RELEASE WORKSPACE
342	CRESDNT	CHAR(1)	A=ALL M=MAN N=NONE
32C	CRESERVE	FIXED(31)	STORAGE
5D8	CRESPMIM	PTR(31)	TEMP RESCH UK LEMP RES
5DC	CRESTORE	PTR(31)	ADDR OF RESTORE ROUTINE
5D6	CRETCODE	PTR(8)	RETURN CODE AREA (0 10)
5E4	CRLSE	CHAR(1)	YES FOR RELEASE OF SORTWK
5E8	CRMERGE	FIXED(31)	MERGE ORDER - REDUCTION
5EC	CRPLIN	PIR(31)	ADDR OF VSAM INPUT RPL
5F4	CRFLUUI	PTR(31)	ADDR OF VSAM OUTFOLKEL
6D4	CRSAFULL	BIT(1)	1=RSA IS FULL
32D	CRSRVD24	PTR(24)	CALLING PROGRAM
140·	CRTHADDR		ADDR. OF PERMANENT RTNS
048	CSAVELL	DTD(31)	LEVEL 1 KUUTINE SAVE AREA
070	CSAVELZ	FIRCOLY	I EVEL 2 ROUTINE SAVE AREA
000	CSAVEOS	PTR(31)	SUPERVISOR AND DM SAVE AREA
690	CSAVEPTR	FIXED(31)	SAVE AREA POINTER
7F4	CSAVER14	FIXED(31)	SAVE AREA IN ALL PHASES
388	CSAVITIK CSAVREG7	FIXED(31)	SAVE INPUT PIR (ICELIM)
60C	CSCANLOC	PTR(31)	ADDR. OF LAST SCAN LOCATION
344	CSECALC	CHAR(1)	Y=AUTOM. SEC. ALLOC.
684 486	CSECALL	CHAR(2)	SECONDARY ALLOCATION SWITCH
442	CSEELKK	CHVED(21)	YES = DO STAND-ALONE SEEKS
618	CSELECT	PTR(31)	ADDR. OF TSELECT ROUTINE
494	CSFROMSW	BIT(1)	FROM-REG IN RCD-MOVE SAVED
495	CSHORTSW	BIT(1)	SHORT VAR RECORD FOUND
474 608	CSKIPPFC	EIXED(31)	NUMBER OF RECORDS TO SKIP
609	CSKIPR24	PTR(24)	THREE BYTES
34B	CSMF	CHAR(1)	S=SHORT SMF RECORD
600	CSORTIN	FIXED(31)	SORTIN DCB POINTER
4 7 Ø 6 N &	CSURINAM	CHAK(4) ETYEN(21)	PREFIX FUR DU STATEMENTS
61C	CSORTWK	PTR(31)	PTR TO SORTWK DCR ITST
5D7	CSPANIN	BIT(1)	ON = VAR. SPANNED INPUT
5B8	CSPNVSAM	PTR(31)	-
5E5 774	CSRTCHTL	BIT(1)	SORTCHTL STATEMENT PRESENT
49C	CSRTMODS CSRTMRGE	PTR(31) CHAR(1)	SORTMODS DCB POINTER SORT=(S)   MERGE=(M)
16C	CSTAADDR	PTR(31)	ADDR OF INIT STAE
34D	CSTAE	CHAR(1)	Y = STAE WANTED AT ABEND
688 689	CSTAESAV	PTR(31)	STAE WORK AREA -
689 6ED	CSTAES24 CSTEPNAM	PTR(24) Char(8)	FOR NEXT LEVEL STEP NAME
7E0	CSTOPIN	CHAR(1)	CANT USE EXCP FOR SORTIN
7E1	CSTOPOUT	CHAR(1)	CANT USE EXCP FOR SORTOUT
494	CSTOSW	BIT(1)	TO-REG IN RCD-MOVE SAVED
614 798	CSTRINGS CSTWNAME	FIXED(31) CHAR(12)	STRING COUNT FOR STOW MACRO
. , 5	STRUME	OHUK(12)	TOK STOW PROMO

Licensed Material--Property of IBM
Page of LY33-8042-6, As Updated 31 March 1981, By TNL LN20-9345

l				
l	DISP	LABEL	ATTR	
	400	ACURDA 4 A		
	480	CSUBCTAT	FIXED(15)	SOR-BIN/RAZE-RIN
	348	CONDOINE	PINEU(13)	SNC INSIDIULIUN
ŀ	490	CSYSDUMP	CHAD(1)	AES = CASHUMME UD CASTBERD
	5 F 8	CSYSIN	PTP(31)	CYCIN DOR POINTED
ŀ	774	CSYSLIN	PTR(31)	SYSLIN DCB POINTER
	784	CSYSLMOD	FIXED(31)	SYSTMOD DCB ADDRESS
	338	CSYSNAME	CHAR(8)	NAME FOR PRINT DD CARD
İ	5FC	CSYSOUT	PTR(31)	SYSOUT DCB POINTER
	620	CSYSOUTD	PTR(31)	SYSOUT - DCB
	694	CSYSRES	FIXED(31)	SYSTEM NEEDS
	4AB	CSYSTEM	CHAR(1)	O=MFT/MVT S=SVS M=MVS
	/88	CTABPTR	PTR(31)	POINTER IN SCANTAB
	494	CIECHNOS	BIT(1)	TECHNIQUE FORCED
	120	CIECHNU	CHAR(1)	PIBIDIXIN OR DEFAULT E
l	120	CTEMP100	LIYED(21)	WURK AREA
	123	CTEMP115	FIKID) ETVEN/1EN	MUKK AKEA
i	121	CTEMP124	DID(34)	MORK AREA
	124	CT FMP2	FIXED(31)	MODE ADEA
	127	CTEMP208	PTR(8)	MORK ARFA
	126	CTEMP215	FIXED(15)	WORK ARFA
	125	CTEMP224	PTR(24)	WORK AREA
	128	CTEMP3	FIXED(31)	WORK AREA
	12B	CTEMP308	PTR(8)	WORK AREA
i	12A	CTEMP315	FIXED(15)	WORK AREA
	129	CTEMP324	PTR(24)	WORK AREA
	120	CIEMP4	FIXED(31)	WORK AREA
	125	CTEMPASE	PIKIBI	WUKK AKEA
	120	CTEMP415	LIVER(12)	MUKK AKEA
	680	CTESTREY(3)	PTP(16)	CIC EDD THO CONTROL HORDS
	1 D 0	CTFCB	FIXED(31)	TECH CHAIN HEAD
	164	CTIME	PTR(31)	ADDR. OF ROUTINE TIMING
	358	CTIOTIN	PTR(31)	PTR TO INPUT DD NAME (DEF)
	59C	CTIOTOUT	PTR(31)	PTR OUTPT DD NAME (DEF-DEG)
	6 A 8	CTMPSIZ	FIXED(31)	TEMP ALLOC SIZE
i	808	CTOTTIME	FIXED(31)	CPU TIME AREA FOR SMF
	495	CTPDYNSW	BIT(1)	NO DYNNUMB FOR TAPE
	828	CIRACE	CHAR(128)	TRACE TABLE
	810	CIREEBAS	F1XED(31)	TREE BASECODE ADDRESS
ŀ	409	CTREELUC	LIK(21)	AUUK, UP SIAKI UP IKEE
	640	CINCESIZ	LIYEN(31)	1000 ALEAD TOEE
	644	CTRESHID	FIXED(31)	EDEE RYTES IN DSA
	158	CTRKROUT	PTR(31)	ADDR. OF ROUTINE TRKROUT
	6BC	CURRDATE	FIXED(31)	CURRENT DATE - PACKED DEC.
1	6B8	CURRDECB	PTR(31)	ADDR. OF CURRENT DECB
l	6C0	CURREC	PTR(31)	ADDR TO BUFFERED REC IN RSA
	6 A D	CUTABENT	PTR(31)	PTR/DISPL TO UNIT TAB ENTRY
	5D7	CVARREC	BIT(1)	ON = VARIABLE LENGTH RECS
ŀ	34C	CVBLKSET	CHAR(1)	N =BYPASS VLR-BLOCKSET
	6C4	CVERACC	FIXED(31)	ACCUMULATOR FOR VERIFY
	345 6D4	CVERIFY CVIM	CHAR(1)	Y=VERIFY OUTPUT RECS 1=BYPASS VED CALL VIM
l	340	CVIO	BIT(1) CHAR(1)	1=BYPASS VED CALL VIM VIO UNDER MVS
	34E	CVIOREC	CHAR(1)	VIO OR TAPE REC
}	6 D O	CVMOVEL	FIXED(31)	VLR - NO OF BYTES TO MOVE
l	6 D 4	CVREMSW	BIT(8)	ICEVRE MISC SWITCH
Ì	6C8	CVSAMERR	PTR(31)	ADDR OF MODULE ICEERR
1	6CC	CVSAMRVE	FIXED(31)	STORAGE RESEVED FOR VSAM
1	496	CVSAMR1	BIT(1)	VSAM RELEASE 1 IN SYSTEM
1	75C	CWBPOOL	FIXED(31)	WB-POOL ADDRESS
	760	CWBPOOLL	FIXED(15)	WRITEBACK POOL SIZE
	762	CMBPOOLC	FIXED(15)	CURRENT POOL SIZE

Licensed Material--Property of IBM Page of LY33-8042-6, As Updated 31 March 1981, By TNL LN20-9345

DISP	LABEL	ATTR	COMMENT
764	CWBPOOLR CWKVAMDS CWMINCON CWMINREC	FIXED(15)	OFFSET TO RESERVE BLOCK
5E5	CWKVAMDS	BIT(1)	VIO WORK DATASET PRESENT CTRL FIELD BASED MIN RECLEN
6 D 6	CMMINCON	FIXED(15)	CTRL FIELD BASED MIN RECLEN
766	CWMINREC	FIXED(15)	MIN EXTRACT IN NON-PEER
758	CWURKQ	PIK(31)	
120	CWORK1	FIXED(31)	
123	CWORK108		
122	CWORK116	FIXED(16)	WORK AREA
121	CWORK124	PTR(24)	WORK AREA Work Area
124	CWORK2 CWORK208	LTYEN(21)	
127	CHORKSIA	PTR(8) FIXED(16)	WORK AREA
126 125	CWORK216	PTR(24)	HILL THEY
128	CWORK224 CWORK3	FIXED(31)	
12B	CWORK308	PINCULITY	WORK AREA
12A	CHOPKSIA	ETYED(14)	WORK AREA
129	CWORK324 CWORK4 CWORK408 CWORK416	PTR(24)	WORK AREA
12¢	CMUSK4	FIXED(31)	LIODY ADEA
12F	CWORK408	PTR(8) FIXED(16)	WORK AREA
12E	CWORK416	FIXED(16)	WORK AREA
12D	CWORK424	PTR(24)	WORK AREA
130	CUNRKS	FIXFD(31)	WORK AREA
134	CWORK6 CWORK7 CWORK8	FIXED(31)	
138	CWORK7	FIXED(31)	WORK AREA
13C	CWORK8	FIXED(31)	WORK AREA
6D5	CWRKTYPE CWTO	CHAR(1)	T=TAPE D=DISK
6 D 8			
178	CXCCOUNT	FIXED(31)	
17C	CXCLNG	FIXED(31)	WORK AREA FOR EXTRACT
170	CXLIST	PTR(31)	ADDRESS OF EXIT LIST
800	CXTRSTAT	PTR(31)	SMF_STATISTICS_RTN PTR
7B8	CZONE61	CHAR(32)	INTERM. STOR. F. ZD W. E61

# Appendix A: Program Exits

# Calling Modules

į	Disk				Tape					Merge				
Exit	BLOC FLR VLR	PEER	VALE	CR fix		BA fix		OS fix	CL var		LY var	BA fix	LN var	†  -only 
Ph0		† 	} 	+ !		} !		 		 		} 1		} 
E18			DED	j -	- 1	j -	-	-	-	j -	-	j -	-	j -
E39 [		DED	DED	! -	-	! -	-	-	-	-	-	-	-	! -
Ph1		1	!	†		r I								1
E11 j		i -	VRE1	APL	APL	APG	APG	APL	APL	APG	APG	APG	APG	ì -
E15 į	E15 <sup>2</sup> 15V <sup>8</sup>	CRE	VRE 1	RDR	RDS	RDD	RDE	RDR	RDS	RDD	RDE	RDD	RDE	j -
E16 j	<b>-</b> -	1 -	VRE1	RDR	RDS	RDD		RDR		RDD		RDD	RDE	<b>i</b> -
E17		-	VRE3	8PM	8PM	RPC	RPC	RPM		RPC		RPC	RPC	-
E18	<b>-</b> -	-	VRE3	9GN		AGI	AGT	AGN		AGA		AGA	AGA	1 -
E19 [		ļ	VRE	9GN	9GN	AGI	AGI	AGN	AGN	AGA	AGA	AGA	AGA	-
E31		[ -	VRO	-	- 1	! -	-	-	-	-	-	-	-	] -
E35		CRO	VRE	-	-	! -	-	! -	-	-	-	-	-	ļ . <del>-</del>
E37		! -	VRO	i -	-	-	-	-	-	-	_	! -	-	! -
E39		-	VRO	-	-	-	-	-	-	-	-	-		! -
E61	 	CRE	J VRE1	ROB	ROD	ROB	ROD	ROB	ROD	ROF	ROH	ROB	ROD	<del>-</del> 
Ph2		i	i	i		i		i	,	i		Í		i
E31	<del>-</del>	1 -	VED	APL	APL	APH	APH	APL		APH		APH	A PH	1 -
E25		-	VEE	RBW	RBX	RBJ	RBK	RBW		RBJ	RBK	RBJ	RBK	-
E27		1 -	VED	8PM	8PM	RPF	RPF	RPM		RPF		RPF	RPF	1 -
E28		1 -	-	9GN	9GN	AGJ	AGJ	AGN		AGG		AGG	AGG	i -
E29		! -	-	9GN	9GN	AGJ	AGJ	AGN		AGG		AGG	AGG	! -
E61		<u> </u>	VEE	ROQ	ROQ	ROQ	ROQ	ROQ	ROQ	ROQ	ROQ	ROQ	ROQ	<u> </u>
Ph3		ĭ	1	, 								 		I
E31		j -	VIP	API	API	API	API	API	API	API	API	<b>JAPI</b>	API	API
E32	i	j -	j -	i -	-	<b>i</b> –	-	i -	-	i	-	j -	-	RDU
E35	E35*35VL	LIMS	VIM	RBM	RBO6	RBM	RBQ 6		RBO6		RBO6	•	RBO <sup>6</sup>	RBM?
E37	j	j -	VIM -	RPG		RPG	RPG	RPG		RPG	RPG	RPG	RPG	RPG
E38	ļ - <b>-</b>	j -	j -	AGK		AGK	AGK	APK		APK		APK	APK	APF
E39		LIM	ATA	AGK		AGK	AGK	APK		APK		APK	APK	APF
E61		-	1 -	ROQ	ROQ	ROQ	ROQ	ROQ	ROQ	ROQ	ROQ	ROQ	ROQ	ROQ

Or VRN
OR IPUM if no SORTIN
OR VRN OR VRO
OR COBU if no SORTOUT

\*8BO for spanned records
\*7RBO for variable-length records
\*9Or COBV if no SORTIN

## Register Usage

The general registers used by the sort/merge program for linkage and communication of parameters follow operating system conventions.

Register 1 is used to pass the address of a parameter list to the called routine.

Register 13 contains the address of an area set aside by the sort/merge program, in which a user routine may save the contents of registers.

Register 14 contains the address of the sort/merge program return point.

Register 15 contains the address of the user routine. It is also used by the user routine as a return code register to communicate information to the sort/merge program.

# Appendix B: Format Codes (Conventional Techniques)

With a conventional sort or merge each node in the tree contains a code describing the current status of the node.

The structure of a node, and the various format codes as they appear in the second word of that node, are described below.

For fixed-length records the node consists of five words. The first word contains the address of the next-level node with which records associated with the current-level node are compared. In other words, the first-level node points to the second-level node, which points to the third-level node, etc.

The second word contains the format code. This code is a number that is used as a displacement value to index a branch table in the ordering module. The entries in the branch table reflect the sequence (old or new) to which each record at a node belongs. This knowledge precludes needless compares and facilitates the updating of a node after the position of a new record is determined.

The last three words refer to the actual addresses of the three records in the RSA. A binary compare is made to determine which word in the node is to receive which RSA address. If the user specifies ascending sequence, the address of the record with the smallest control field is placed into the first word, the address of the record with the largest control field is placed into the third word, and the address of the record with the control field that collates in the middle is placed into the second word. For descending sequences the order is reversed.

For variable-length records, the node consists of only three words. The first two words contain the next-level node address and format code and are functionally similar to the fixed-length record node described above.

Because of the complexity of address structuring in the variable-length record format, only one record is referred to in the RSA. Hence, only one word is required in the node for this purpose. Functionally, however, the word is similar to that in the fixed-length record node.

# Fixed-Length Records

The format codes for fixed-length records are interpreted as follows:

Code	Meaning					
0	No record addresses in node.					
16	An event has occurred: flushing completed, winner obtained, or new string started. This is a program node, as opposed to a tree node.					
32	One address in the node. Record is for new sequence.					
48	One address in the node. Record is for same sequence.					
64	Two addresses in the node. Both records for new sequence.					
80	Two addresses in the node. One record for new sequence, and one record for same sequence.					
96	Two addresses in the node. Both records for same sequence.					
112	Three addresses in the node. All records for new sequence.					
128	Three addresses in the node. Two records for new sequence, one record for same sequence.					
144	Three addresses in the node. Two records for same sequence, one record for new sequence.					
160	Three addresses in the node. All records for same sequence.					

# Variable-Length Records

For variable-length records, a slight difference occurs in the format codes because only one record address is entered in each node. The various codes and their meanings are as follows:

Meaning			
No record address in the node.			
An event has occurred: flushing completed, winner obtained, or new string started. This is a program node, as opposed to a tree node.			
Describes the status of the node (see program listing for details).			
Record address in the node is for a new sequence.			
Describes the status of the node (see program listing for details).			
Record address in the node is for the same sequence.			

#### FORMAT CONDITION CODES

The format code determines the point of entry into the instruction sequence. When the instruction sequence is entered, one of four condition codes exists. These condition codes dictate the final disposition of the record and are as follows:

Condition	Interpretation
Flush	Forces records from the tree.
Fill	Continue filling the tree.
Same	Record is of same sequence as previous records.
New	Record begins a new sequence.

#### Example: Fixed-Length Records

The examples below will aid in interpreting the listings for fixed-length records. The character to the left of the slash represents the record entering the node, and the three characters to the right of the slash represent the records already in the node. Hence, in the program listing, when the comments contain a statement 'This case handles an x/xxx situation', it can be resolved as follows:

Condition	Interpretation
s/sss	New record is of same sequence as three previous records.
n/sss	New record begins new sequence; previous three records of same sequence.
-/sss	Records in tree are of the same sequence, and the program is in flush mode.
n/SSN	Two sequential records, one new sequence in node. New sequence record entering.
-/NNN	Flush mode; all records of new sequence to be flushed.
-/ss-	Flush mode; two records of same sequence to be flushed.
N/S	One record in the node is of the same sequence; new record begins a new sequence.

### Example: Variable-Length Records

For variable-length records, the listings may be interpreted as explained below. The two characters to the left of the slash represent the record entering the node and the two characters to the right of the slash represent the record already in the node.

Condition	Interpretation
S1/S1	New record is of same sequence as previous record.
T1/S1	Record in node is of the same sequence as others in tree. Set status of node to reflect this condition.
T2/S1	Record in node is of the same sequence. New record will change sequence. Set status of node to reflect change.
S2/S2	Record in node is of new sequence. New record is also for new sequence.
T2/S2	Record in node is for new sequence. Set node status to temporary new sequence record.
T2/T1	Set node status at temporary before next record is introduced.

# Appendix C. Checkpoint/Restart Facility

If it is included at system generation the sort/merge program may make use of the Checkpoint/Restart Facility of the operating system. The user directs the sort/merge program to use this facility by (1) including the checkpoint parameter (CKPT) in the SORT or MERGE control statement, and (2) providing a SORTCKPT DD statement to define the checkpoint data set; this data set will store the information needed to restart processing.

| Checkpoint procedures are described below for Peerage and Vale, and for | the conventional sort/merge techniques.

| With Blockset checkpoints are not taken. If a checkpoint has been | requested the Blockset technique is rejected and another used instead.

## Sorting Application (Peerage and Vale)

If checkpoints are requested, the checkpoint macro (CHKPT) is issued and checkpoint records are written on the checkpoint data set as follows:

- ICECKP/ICEVED start of intermediate merge phase
- ICEVIM start of final merge phase
- when the checkpoint track pool is empty.

The relative track addresses of 20-30% (maximum 1364 tracks) of the tracks primarily allocated to the intermediate work files are saved in a track pool. Every time a track of blocks has been read, the relative address is saved in the track pool. When the blocks have to be written the address of an unused track is found in the track pool. A checkpoint is taken when there are no more unused tracks. After the checkpoint has been taken, the saved tracks are available for use again.

## Sorting Application (Conventional Techniques)

If checkpoints are requested the following modules link to module ICECHK. This issues checkpoint macro instruction (CHKPT), which causes checkpoint records to be written on the checkpoint data set.

- ICEAPC Start of phase 1 (all techniques)
- Start of phase 2 (all techniques except crisscross and **ICEAPJ** oscillating) Start of each intermediate merge phase pass (balanced direct access technique)
- During phase 2 (oscillating technique) **ICERON**
- **ICEROS** During phase 2 (polyphase technique)
- ICEROT During phase 2 (balanced DASD technique)
- ICEAGH Start of phase 3 (all techniques)

The program can be restarted from the checkpoint taken at the start of the sort phase, or from the last checkpoint taken.

## Merge-Only Application

If checkpoints are requested they are taken whenever an end-of-volume condition occurs on SORTOUT, unless the user provides his own exit list address for the SORTOUT DCB. Module ICEAPF inserts the address of the checkpoint module ICE8CR in the DCBEXLST field when it generates the SORTOUT DCB (unless this field has been set by a user routine at exit E39). Module ICE8CR is thus entered at the DCB exit list, which contains only the end-of-volume entry. This entry points to the EOV exit routine in this module, which in turn issues a checkpoint macro instruction (CHKPT), and then returns to the IOCS.

# Appendix D. Program Listing Standards and Conventions

To facilitate the identification of modules, work areas, tables, and other aspects of the sort/merge program listing, each technique assigns symbolic names according to a pattern.

#### **Blockset**

#### MODULE NAMES

| The format of all module names except ICEMAN and ICEAM1 is ICEmod, where:

ICE is the identification for program modules

mod is a descriptive name for the module (four characters)

#### INSTRUCTION NAMES

| Descriptive names are used. All variables, which are resident in | COMMON, begin with the characters 'COM'.

#### CONSTANT NAMES

| Descriptive names are used. 'Constant' is used to mean any item of data | defined outside COMMON which is not dynamic (i.e. based on a pointer).

#### WORK AREA NAMES

| Since the program is reenterable, all work areas are variables, and are | thus resident in COMMON

## Peerage and Vale

#### MODULE NAMES

| The format of all module names is ICEmod, as described under 'Blockset' | above. The descriptive portion (mod) is 3 characters long.

#### INSTRUCTION NAMES

Descriptive names are used. All variables, which are resident in ICECOMMA, begin with the character 'C'.

#### CONSTANT NAMES

Descriptive names are used. 'Constant' is used to mean any item of data defined outside ICECOMMA which is not dynamic (i.e. based on a pointer).

#### WORK AREA NAMES

Since the program is reenterable, all work areas are variables, and are thus resident in ICECOMMA.

## Conventional Techniques

#### MODULE NAMES

! The format of module names is ICEtmm, where:

ICE is the identification for program modules.

t, in general, is either an 'A' for an Assignment or an 'R' for a Running type module; however, for some assignment modules associated with the crisscross technique, t is '9', and for some running modules associated with the crisscross technique, t is '8'.

mm is the unique portion of the module name.

#### INSTRUCTION NAMES

The format of all internal type instruction names is mmnnnnno or mmmnnnno where:

mm or mmm represents the last two or three characters of the module name.

nnnnn or nnnn is a unique designation assigned by the programmer and may be from one to five characters.

o is an 'X' if the label is externally used; otherwise it can be used as another n.

# Appendix E: SMF Record DSECT (ICESMF)

This area is allocated and filled in by ICEEXIO, ICEOXOV, or ICEMOS depending on the sort technique used. The record is passed to sort's SVC routine, ICEFIXM (OS/VS2) or ICEFIX (OS/VS1). The SVC routine uses the passed record and issues the SMF macro to write the SMF record out to the SMF data set, upon successful completion of the sort/merge application.

```
DESCRIPTION
DISP NAME
 ICESMF DSECT
0000
 RECORD DESCRIPTOR WORD
 ICERDW
 XL4
0000
 DS
 ICESIND DS
 SYSTEM INDICATOR
BITO: SUBSYSTEM ID FOLLOWS SYSTEM ID
0004
 В
 SMF RECORD TYPE. (16)
TIME RECORD WAS MOVED TO SMF BUFFER
DATE RECORD WAS MOVED TO SMF BUFFER
 ICERTYP DS
0005
 ICEBTIME DS
ICEBDATE DS
0006
 FL4
 PL4
000A
 SYSTEM IDENTIFICATION
 CL4
000E
 ICESID
 DS
0012
 ICEJOBNM DS
 CL8
 JOBNAME
 TIME READER RECOGNIZED JOBCARD DATE READER RECOGNIZED JOBCARD USER ID (TAKEN FROM COMMON EXIT PARAMETER AREA)
STEP NUMBER
001A
 ICERST
 DS
 FL4
 ICERDS
001E
 DS
 PL4
0022
 ICEUIF
 CL8
 DS
 ICESTN
002A
 DS
 XL1
 RESERVED
002B
 ICERES1
 DS
 CL3
002E
 ICESUBID DS
 CL4
 SUBSYSTEM ID
 XL2
 RECORD SUBTYPE
SHORT RECORD
FULL RECORD
 DS
0032
 ICERSUB
 X'01'
X'02'
 ICERSUBS EQU
ICERSUBF EQU
 SELF DEFINING SECTION
 ICEPROD DS
 OFFSET TO PRODUCT SECTION
PRODUCT SECTION LENGTH (10)
NUMBER OF PRODUCT SECTIONS (1)
0034
 F
0038
 Н
 ICEPRODN DS
003A
0030
 ICEDATA DS
 OFFSET TO SECTION COMMON TO SHORT AND FULL RECS
 ICEDATAL DS
ICEDATAN DS
 DATA SECTION LENGTH (46)
NUMBER OF DATA SECTIONS (1)
0040
 н
0042
 Н
0044
 ICESTAT
 DS
 OFFSET TO RECORD LENGTH STATISTICS
 ICESTAT DS
 STATISTICS SECTION LENGTH (64)
0048
 ICESTATH DS
 NUMBER OF DATA SECTIONS (110)
004A
 Н
 PRODUCT SECTION
 ICERECV DS CL2
ICEPRDCT DS CL8
 RECORD VERSION. '50' FOR RELEASE 5.0 PRODUCT NAME. '5740-SM1'
004C
004E
 DATA SECTION. COMMON PART
 ICECDAT
 START OF DATA SECTION
TO PUT NEXT FIELD ON FULLWORD BOUNDRY
0056
 DS
 0 H
0056
 ICERES2
 DS
 CL2
0058
 ICESTPHM DS
 STEPNAME
 CL8
 ICERCDS DS
ICEBYTES DS
 NUMBER OF RECORDS SORTED
NUMBER OF BYTES SORTED (SUM OF RECORD LENGTHS)
SORT CPU TIME, HUNDREDTHS OF A SECOND
0060
0064
0068
 ICECPUT
 DS
 F
006C
 ICELEN
 DS
 SPECIFIED RECORD LENGTH
006E
 ICEIBLK
 DS
 INPUT BLOCKSIZE (MAX)
 OUTPUT BLOCKSIZE
TOTAL CONTROL FIELD LENGTH (NUMBER
OF BYTES ACTUALLY COMPARED BY SORT)
0070
 ICEOBLK
 DS
 н
 ICEKEYLN DS
0072
```

# Licensed Material--Property of IBM Page of LY33-8042-6, Added 31 March 1981, By TNL LN20-9345

```
DISP NAME
 DESCRIPTION
 NUMBER OF WORK DATA SET TRACKS USED BIT 0: RESERVED BIT 1-2: =FIXED LENGTH RECORDS
0074
 ICEWBLK DS
 F
0078
 ICEFLBYT DS
 В
 01=VARIABLE LENGTH RECORDS
10=VARIABLE LENGTH SPANNED RECORD
 BIT 3-4: =BLOCKSET
 01=PEERAGE
 10=VALE
 11=CONVENTIONAL AND MERGE TECHNIQUES
BIT 5: '1'B IF SORT DYNAMICALLY INVOKED
BIT 6-7: RESERVED
 NUMBER OF DYNAMICALLY
ALLOCATED WORK DATA SETS
0079
 ICENDYNA DS
 AL1
 ICERES3 DS
007A
 CL2
 RESERVED
 ICEVAR
 EQU ICECTR BEGINNING OF VARIABLE PART
 ICECTR
007C
 DS
 16F
 RECORD COUNTERS FOR INTERVALS 1 - 16
 ICECTR
OOBC
 ORG
 ICECTRO1 DS
 RECORDS IN INTERVAL 1
RECORDS IN INTERVAL 2
007C
0080
 RECORDS IN
RECORDS IN
RECORDS IN
 ICECTR03 DS
 INTERVAL
0084
8800
 ICECTR04 DS
 INTERVAL
008C
 ICECTR05 DS
 INTERVAL
 ICECTRO6 DS
ICECTRO7 DS
 RECORDS IN
RECORDS IN
0090
 FFFFF
 INTERVAL
0094
 INTERVAL
0098
 ICECTRO8 DS
 RECORDS IN
 INTERVAL
009C
 ICECTR09 DS
 RECORDS
 IN
 INTERVAL
ODAO
 ICECTR10 DS
 RECORDS IN
 INTERVAL 10
00A4
00A8
 ICECTRII DS
ICECTRI2 DS
 RECORDS IN
RECORDS IN
 INTERVAL 11
 F
 FFFF
 INTERVAL 12
 RECORDS IN INTERVAL 13
RECORDS IN INTERVAL 14
RECORDS IN INTERVAL 15
RECORDS IN INTERVAL 16
 ICECTR13 DS
ODAC
00B0
 ICECTR14 DS
00B4
 ICECTR15 DS
 ICECTR16 DS
00B8
 ICESMFND EQU
 END OF RECORD
```

# Index

A	codes format 222
appreviations used (MO diagrams) 11	collation 219 COMMA 10
Abend 172	COMMON 10
ABEND dumps 177	communiaction areas
access method (see BSAM, QSAM, VSAM)	Blockset technique
adding	(see COMMON)
temporary change maintenance area 164	conventional techniques
address	(see CPI, PPI)
of DCB table (see PPI index)	Peerage and Vale techniques
of disk directory (see PPI index)	(see COMMA) COMTRACE 178
of GETMAIN tables (see PPI index) of IOB tables (see PPI index)	contents of a specially formatted dump 188
of unit count table (see PPI index)	control blocks 168
node 221	conventional technique
all other techniques 191	module names 227
alphabetically names 111	modules 94
ALTSEQ statement 5	conventional techniques 89
AMASPZAP 164	sorting application 2
APAR 163	condition codes format 222
assembler language	considering located error 161
initiating from 5-6	constant names 226
assignment modules, list of ATTACH 5	control fields 222-223 word 222-223
example 6	word 222-223 CPI 10
example	creation of 22
	CRCX 173,89
	(see crisscross technique)
B	crisscross technique
	distribution sequence 77,79
B (sort blocking) (see PPI index)	criteria for disk techniques 3
balanced disk technique	cross-reference tables 196
distribution sequence 77,79	CPI-PPI 196
balanced tape technique	COMMON, Blockset 204
distribution sequence 77,79	COMMA, Peerage/Vale 209 CSECTS
BALN 173 Blockset 90	name 112
conditions for use 2.1	list of 113
directory 109	Blockset technique 109
layout 104	Peerage/Vale 111
module names 226	conventional techniques 113
modules 92	CTRACE 178
object modules 109	CTRx 174
technique 190,89,103	
blocking (see PPI index)	D
BSAM 1,172	
BUFFER 172	data
size (see PPI index)	areas 9 (see also COMMON, COMMA,
bypass	CPI, PPI) sets 8
C	DCB table address (see PPI index)
	DD statements 19
<del></del>	deactivation 7
calling modules 218	DEBUG statement 5,172
capacity exceeded 71	control statement 172
checkpoint/restart 1	debugging aids 159
facility 224 CLOCK	defaults assumed for program options 157
(Only Peerage/Vale) 174	defaults, generated 157
COBOT. 5	defining problem cause 159

Licensed Material--Property of IBM
Page of LY33-8042-6, As Updated 31 March 1981, By TNL LN20-9345

definition phase (see Phase 0) diagnostic messages 177 direct access technique (see balanced disk technique and crisscross disk technique) directories 109 directory of Blockset technique 109 of conventional techniques 112 of Peerage/Vale techniques 111 disk devices disk directory address (see PPI index) disk sort techniques 2 dump 172 specially formatted 187 dump Peerage 169 169 dump Vale dumps 178

# Ε

END index 41 END statement equals routines (multiple compare) error conditions 85 messages 193-195 eight-byte module name 190 event table Blockset 180 Peerage/Vale 184 EXCP EXEC statement 5 exit name table 21,71 exits 4,5 calling modules register usage E15 routine 6 E35 routine 6

# F

facility checkpoint/restart 77 Fibonacci numbers file size 19 (see also PPI index) final merge phase (see Phase 3) object module in storage dump 190 fixed-length records format codes FLAG (only Peerage/Vale) FLR-Blockset conditions for use 2.2 directory of object modules 109 load module interface 86 modules 94 forcing specially formatted dump 187 format codes 221 condition codes 222 event record 178 of dumps 178

formatted Jump contents 188 interpreting 189



G (numbers of records in RSA)
(see PPI index)
general register 219
generated defaults 157
GETMAIN 71,79,83
tables, addresses (see PPI index)



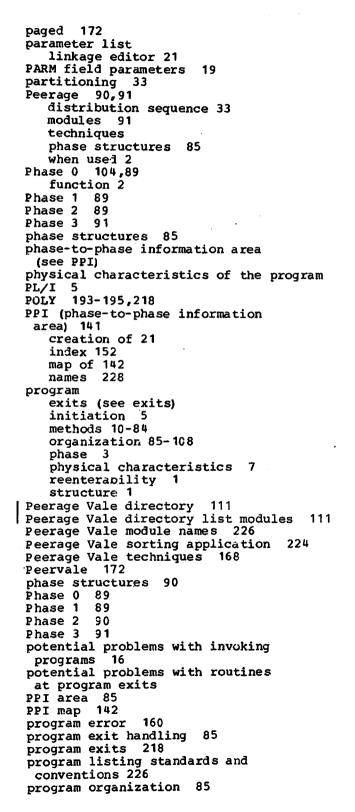
hash table 47 high index (see end index) HMASPZAP 165



ICEAM1 ICECRE 90 ICEIPUT 85 **ICEMESI** 85 ICEMESO 85 ICEMON 85,90 ICEPAR 90 **ICERCB** 85 **ICERCV** 85 ICERED 90 IMASPZAP instruction names 226 interpreting formatted dump 189 index creation of 41 queue 51,53,55 size 19 sorting 43,45,47 information areas conventional techniques (see CPI, PPI) initializing the tree 65 program 17,21 initiating the program 5-6 input pin size (see PPI index) input/output intermediate storage 1 recovery management support 1 interface of load modules intermediate storage I/O operations IOB tables, address (see PPI index) ITT (see index format)

	MO diagram 112 MO diagrams 14-84
	how to use 10
Libraries	MODS 85
system 7	MODS statement 5
private 8	module
LINK 3	Blockset 94
linkage editor 1, 21	conventional 96
parameter list for 22	interface 85-88
LIST 109	Peerage 92
load module Blockset technique 94	tables, how to use 12
conventional technique 96	Vale 93
interface 86	module interface 85 module name 190,226
Peerage technique 92	multiple compare routine 12
structure 91	multiple console support 1
Vale technique 93	marcipio conocio cappara
locating	
control blocks 171	N
SORTWK control blocks in a dump 171	
logical	
block 41	names alphabetic 111
strings 41 low index (see start index) 41	network table 83
low index (see start index) 41	NOABEND 173
<u></u>	NODUMP 173
M	
	0
	U
M (merge order) 21	
(see also PPI index)	object module in
	a dump start 190
	storage dump 190
macro	operating system, relationship to 1
used to initiate the program 5-6	operational
main storage 7 available 21	considerations 5
layout 105-108	methods 10-84
requirements 7	optimization phase (see Phase 0) OPTION statement 5
map index (see mapping)	origin of program messages 190
map of PPI 142	OS (see operating system)
map of data areas (see	OS/VS (see operating system)
COMMON, COMMA, PPI)	OSCL 89
mapping indexes 47	other techniques 192
maximum M 90	overlay load modules 97
diagrams 14-84	output
MAXLIM 7	buffer (see PPI index)
MERGE statement 5 MERGE only application 225	size (see PPI index)
merging technique 225	overlay structure 97
messages	overview of load module interface 85
cross-reference tables 192	MO diagrams 14
origin of 191	conditional techniques sort/merge 75
messages produced using	Blockset technique sort 25
DEBUG control statement 1	Peerage and Vale techniques sort 21
DIAG option 176	program function 15
method of operation 10-84	program phases 4
michrofiche	program structure 2
entry into 10 load module ICEMESI 164	
numbering 11	
organization 164	







queue (see index queue) QSAM 1,71,83

register usage 219

reporting a problem 163 restart 224

technique)

counter (see PPI index) record contents 161 RECORD statement 5 records contents 161 fixed-length (see fixed-length records) number of (G) (see PPI index) spanned (see spanned records) variable-length (see variable-length records) reenterability re-IPL 165 register 1 219 register 13 register 14 219 219 register 15 219

registers used by peerage/vale techniques 165

replacement-selection (see sorting

routines at program exits 160 running modules, list of 112-119



sample set of messages 160 segments 111 SIZE 161 SIZE operand (see PPI index) skip record count (see PPI index) SMF record DSECT 228.1 SMF record type 16 30-31,58-59,89,91 sort invoked via JCL 163 sort uses registers 165 sort/merge general register 219 SORT 5 blocking (see PPI index) SORT generation 7
SORT phase (see phase 1) SORT statement SORTIN control blocks 166,169 sorting application conventional techniques 2 peerage vale 224 sorting techniques 2 SORTMODS 21.5 SORTOUT 102 SORTOUT control blocks 169,166

sortwk 102 sortwk control blocks 166 space 161 specially formatted dump for sort 188 start module in a dump 190 start index 41 structure of the program 1 submitting an APAR 163 submitting incident report 163 symbols used in MO diagrams 10 syntax 173 of buffer (see PPI index) of data set (see also PPI index) of file (see PPI index) SYSABEND 127 SYSIN 21,5 SYSLIN 21 SYSUDUMP 127



tables
names 196
tape
devices 1
table (see PPI index)
tape sort techniques 2
TBLPH1RN 71
TRLPH2RN 79
TBLPH3RN 83
techniques all other 192
trace records 178
trace table 178
transposing records 21
tree updating 41



unit count table address (see PPI index) use of registers 160 user exits (see exits) user routines 5



Vale 90 Vale modules 93 Vale technique 90 distribution sequence phase structures 85 when used 2 variable-length records variable-length records completion codes 221 format codes 223 nodes 223 tree structure 41 VIO 19 VLR-Blockset conditions for use 2.2 directory of object modules 109 load module interface 86.1 modules 95 VSAM 1,39,71,83



WIOB control blocks 172 work area names 226 write chain 41 writeback 51,53,59 criteria for 51

### **READER'S COMMENT FORM**

OS/VS Sort/Merge Logic LY33-8042-6

We would appreciate your comments. Please give specific page and line references where appropriate. Your comments will be carefully reviewed by the persons responsible for writing and publishing this material.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

If you wish a reply, please be sure to include your name and address.

#### **COMMENTS**

Reply request	ed:	Name:
Yes		Job Title:
No		Address:
Data		Zin

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments, or you can send them direct: *IBM Corporation*, P. O. Box 50020, Programming Publishing, San Jose, California 95150, U.S.A.

THANK YOU FOR YOUR COOPERATION
PLEASE FOLD ON TWO LINES, STAPLE AND MAIL

# OS/VS Sore Merge Logic

Cut or Fold Along Line

#### Reader's Comment Form

Your comments about this publication will help us to produce better publication for your use. Each reply will be carefully reviewed by the person responsible for writing and publishing this material. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Note: Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

Fold and tape

Please Do Not Staple

Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES



# **BUSINESS REPLY MAIL**

FIRST CLASS

PERMIT NO. 40

ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Department 825
1133 Westchester Avenue
White Plains, New York 10604



Fold and tape

Please Do Not Staple

Fold and tape



International Business Machines Corporation Data Processing Division 1133 Westchester Avenue, White Plains, N.Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N.Y., U.S.A. 10591

IBM World Trade Europe/Middle East/Africa Corporation 360 Hamilton Avenue, White Plains, N.Y., U.S.A. 10601



This Newsletter No. LN

LN20-9345

Date

31 March 1981

Base Publication No.

LY33-8042-6

File No.

S370-33 (OS/VS)

**Prerequisite Newsletters** 

LN20-9329

#### OS/VS Sort/Merge Logic

© Copyright IBM Corporation 1973, 1979

This technical newsletter, a part of Release 5 of Program Number 5740-SM1, provides replacement pages for the subject publication. These replacement pages remain in effect for any subsequent OS/VS releases unless specifically altered. Pages to be inserted and/or removed are:

```
Cover - ix
1-7 (2.1 and 2.2 added, 8 deleted)
17 - 20
23 - 32
57 - 60
67,68
85 - 96 (86.1 and 94.1 added)
109, 110
121 - 138
157, 158
161, 162
171, 172
179 - 184
189 - 192
195, 196
203 \div 218
228.1 - back cover (228.1 and 228.2 added)
```

#### **Summary of Amendments**

Release 5 of 5740-SM1 provides improved performance, 3375 direct access storage device support, and usability enhancements.

Note: Please file this cover letter at the back of the publication to provide a record of change.

IBM Corporation, P.O. Box 50020, Programming Publishing, San Jose, California 95150



This Newsletter No. LN20-9390 Date

29 April 1982

Base Publication No.

LY33-8042-6

File No.

S370-33 (OS/VS)

**Prerequisite Newsletters** 

LN20-9329 (obsolete)

LN20-9345

## OS/VS Sort/Merge Logic

© Copyright IBM Corp. 1973, 1979

This technical newsletter, a part of Release 5, of OS/VS Sort/Merge, Program Product 5740-SM1, provides replacement pages for the subject publication. These replacement pages remain in effect for any subsequent releases unless specifically altered. Pages to be inserted and/or removed are:

cover, edition notice 1-2.2

Each technical change is marked by a vertical bar to the left of the change.

#### **Summary of Amendments**

This newsletter contains information about the IBM 3880 Models 2, 3, and 13 Control Units.

Note: Please file this cover letter at the back of the publication to provide a record of change.



International Business Machines Corporation Data Processing Division 1133 Westchester Avenue, White Plains, N.Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N.Y., U.S.A. 10591

IBM World Trade Europe/Middle East/Africa Corporation 360 Hamilton Avenue, White Plains, N.Y., U.S.A. 10601